財務データ抽出システムの再構築 ： NEEDS企業財務データを中心に

財務データ抽出システムの再構築
—NEEDS 企業財務データを中心に—

地　道　正　行

要　旨

関西学院大学商学部では，2010年から学内向けに財務データ抽出サービスが行われており，このサービスを司るシステムは，NEEDS 企業財務データをデータベース化することによって設計・実装されている．このシステムによるサービスの運用開始から約10年が経過し，様々な課題があることがわかってきた．本稿では，これらの課題に対する対策を検証し，システムを再構築する．

キーワード：リレーショナル・データベース管理システム（Relational Database Management System），NEEDS 企業財務データ（NEEDS Corporate Financial Data），データ抽出システム（Data Extraction System），前処理（Preprocessing），再現可能性（Reproducibility）

I　はじめに

近年，「データサイエンス」や「ビッグデータ」といった用語が日常的に使用されるようになり，ビジネスに関する意思決定にも「データ」によるエビデンスが要求されるようになってきている（例えば，Taddy, 2019 を参照されたい）．このような用語が顕著に聞かれるようになったのは，2010年の前後であり，日本でも新聞・テレビなどのマスメディアにおいて連日取り上げられるようになったことは周知のことであろう．

このような動向と期を同じくして，地道（2010-a, b），地道（2012）では，

財務データ抽出システムの構築について議論されている．このシステムは，NEEDS 企業財務データ（一般事業会社，金融関連会社）をデータベース化することによって設計・実装されている．このシステムによるサービスの運用開始から約10年が経過し，筆者が研究や講義・演習に利用する中で，以下のような課題があることがわかってきた：

（P1）システム名称の再考
（P2）リレーショナル・データベース管理システムの見直し
（P3）NEEDS 企業財務データの仕様変更への対応
（P4）抽出システムのインターフェースデザイン改良
（P5）財務データの拡充
（P6）ダウンロード法の拡充

本稿では，これらの課題に対する対応策を検討し，このシステムを更改するために実験環境を構築し，検証した結果について述べる．本稿の構成と課題との対応は以下のようなものである：

Ⅱ節：課題（P1）
Ⅲ節：課題（P2）
Ⅳ節：課題（P3）
Ⅴ節：課題（P4），（P5），（P6）

なお，付録 A にはリレーショナル・データベースに関する用語などの説明を与えており，付録 B には，2019年版の NEEDS 企業財務データ（一般事業会社）の解説と，それにもとづくデータベースの構築について詳細に述べている．また，付録 C には，再構築した企業財務データ抽出システムの詳細を与えている．

Ⅱ　システム名称の再考

課題（P1）としてとりあげた従来のシステム名称 “KGUSBADES”（Kwansei Gakuin University, School of Business Administration, Data Extraction System の略）の「欠点」としては，このシステムを紹介するときなどに，発音

が難しいという指摘がなされることが多かった．また，綴りが長いため覚えにくいという指摘もあった．これらのことから，今回の再構築に際して，システム名称をなじみやすいものにするために改名することを検討した．検討の過程で，大学名のイニシャル K(wansei) から始めると読みにくかったり，データ抽出システムを表す D(ata) から始めても同様に語呂が悪いといった印象があった．様々な検討の結果，新名称を “SKWAD”（スクワッド）とすることにした．これは，変則的ではあるが，School of business administration, KWAnsei gakuin university, Data extraction system の略である[1]．あくまでも印象であるが，旧システム名称よりも簡単に発音でき，綴りも短いため，覚えやすいものと思われる．

Ⅲ　リレーショナル・データベース管理システムの見直し

ネットワーク上でデータベースを利用したオーソドックスなシステム構成としては，Ubuntu (Linux)，macOS といった UNIX 系オペレーティングシステム（Operating System: OS）上でリレーショナル・データベース管理システム（または「関係データベース管理システム」）(Relational Dababase Management System: RDBMS）と Web サーバとして Apahche HTTP Server, 汎用スクリプト言語である PHP[2] を利用したものである．

OS として，macOS，Ubuntu を利用し，RDBMS として MySQL と PostgreSQL を選択した場合のシステム構成は以下のような表にまとめられる：

表 1 ：MAMP, MAPP, LAMP, LAPP 環境

OS \ RDBMS	MySQL	PostgreSQL
macOS	MAMP	MAPP
Ubuntu	LAMP	LAPP

1） SKWAD（スクワッド）は，精鋭（特殊）部隊という意味の単語 squad の発音（skwa(:)d）と同じであり，「選りすぐりの財務データ」の一団を表すという理解もできる．
2） https://www.php.net/

例えば，LAPP の場合は，OS として Ubuntu（Linux 系 OS）上で，Web サーバ Apache HTTP Server（`httpd`）と RDBMS である PostgreSQL[3] を導入し，PHP でそれらを連携しながら運用することを意味している．各システム構成の概念図を図 1 に与える．

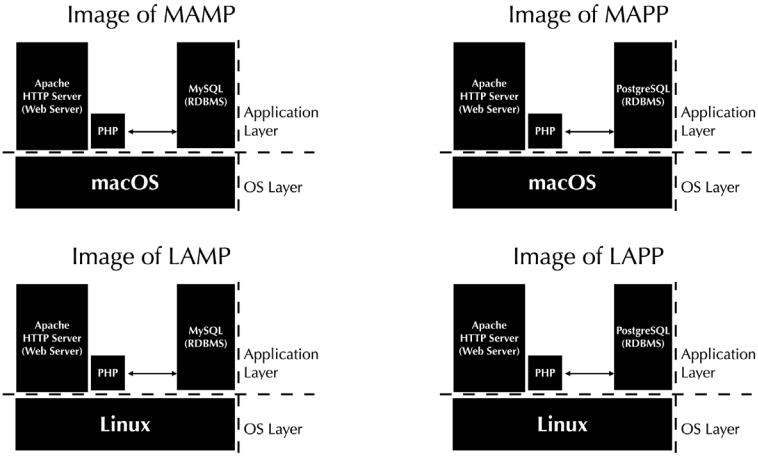


図 1 ：MAMP，MAPP，LAMP，LAPP 環境のイメージ

課題（P2）に関しては，従来のシステムでは RDBMS として MySQL[4] を利用してきたが，MySQL を取り巻く最近の動向[5] を勘案すると，データ抽出システムを PostgreSQL をベースに再構築することが，環境の変化に対する頑健性を確保するために必要と考えられた[6]．このような理由から，本稿

3） `https://www.postgresql.org/`
4） `https://www.mysql.com/`
5） 商用データベースを販売する業界トップ企業である Oracle 社（`https://www.oracle.com/`）によって（2009年から2010年に）Sun Microsystems 社が買収されたことにより，MySQL は Oracle 社によって開発が続けられている．なお，Sun Microsystems 社で開発された Java 言語もこの買収の際に移管され，現在に至っている．
6） RDBMS の選択を再検討したもう一つの理由が，MySQL のバージョン間での仕様変更である．筆者だけかもしれないが，MySQL のバージョンアップ（例えば，バージョン5.6からバージョン5.7）に伴う仕様変更には戸惑いを覚える．筆者は利用したことがないが，MySQL からのフォークである MariaDB の導入も検討に値すると思われる．なお，MySQL については，たとえば，西沢（2017）を，PostgreSQL については，鈴木（2012）を参照されたい．

では以下のような環境を実験的に構築した[7]：

Specification	MAMP	MAPP	LAMP	LAPP
Hardware	MacBook Pro 2018	MacBook Pro 2018	仮想環境	仮想環境
OS	macOS Catalina（10.15.6）	macOS Catalina（10.15.6）	Ubuntu（18.04）	Ubuntu（18.04）
CPU cores	6	6	4	4
Main Memory	32 GB	32 GB	8 GB	8 GB
RDBMS	MySQL 5.6	PostgreSQL 12.2	MySQL 5.7	PostgreSQL 12.4

なお，CPU は Intel(R)Core(TM)i9, 2.9 GHz であり，仮想化ソフトウェアとしては VMware(R)[8] Fusion Pro（バージョン11.5.6）を利用している．

Ⅳ　NEEDS 企業財務データの仕様変更への対応

課題（P3）は，2010年代初頭の NEEDS 企業財務データから，その構造がそれまでのものと全く異なったものとなったことへの対応である．それ以前は，連結決算，単独決算でデータファイルが分かれており，会計基準も一種類（日本基準）であったが，変更後は連結決算と単独決算が合併され，３種類（日本基準，米国会計基準，国際会計基準）の会計基準が存在し，本決算に加えて，四半期決算のデータも収録されるようになった．なお，これらの変更にともなって，データの一意性を保証するための主キーが，２種類（日経会社コード，決算年月日）から，５種類（日経会社コード，連結基準フラグ，決算年月，決算種別フラグ，レコード種別）に増加した．データベースを再構築するにあたっては，これらの変更に対応する必要があるとともに，データ量の増加に伴って，抽出時間を短縮することを考慮しつつ，データベースからデータを抽出する方法を再検討する必要があった．

これらの問題に対して，以下のような方法でデータベースを構築した：

7）実際の学内サービスは，別途サーバ環境を構築して，2021年度から運用する予定である．

8）https://www.vmware.com/

（DB1）データの前処理のためのスクリプトを新しいデータファイルの形式へ対応させるために修正[9)]

（DB2）MySQL と PostgreSQL の両方でデータベースを構築

（DB3）全データを使って構築したデータベースから，連結本決算と単独本決算のデータベースを分離・構築

なお，データファイルとしては2019年に納品されたものを利用して，作業工程毎のスクリプトを作成したものを `Makefile` にまとめ，作業工程を `make` コマンドによって自動実行する仕様とした．このことは，地道（2012）でも指摘されているが，データベースを迅速かつ正確に自動的に構築したり，サービスの拡大を行うために必要となったためである．

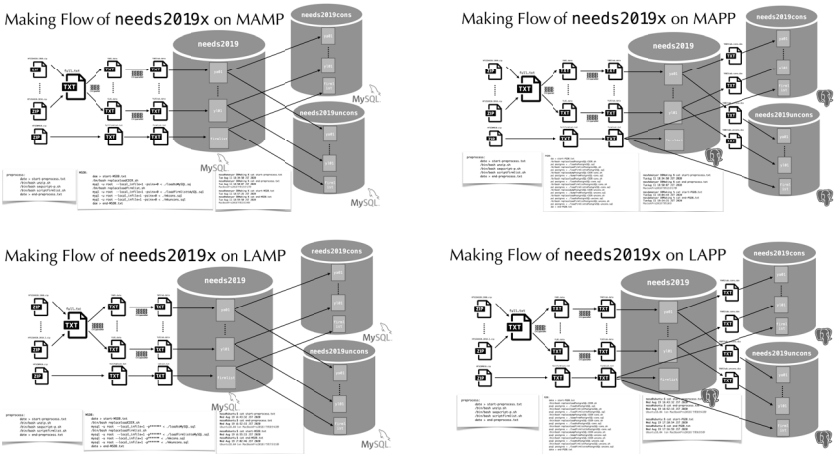


図 2 ：MAMP，MAPP，LAMP，LAPP 環境のもとで NEEDS 企業財務データ（一般事業会社）2019年版に基づくデータベース構築のイメージ

図 2 には，MAMP，MAPP，LAMP，LAPP 環境のもとで NEEDS 企業財務データ（一般事業会社）2019年版に基づくデータベース構築のイメージを

9） 前処理には，GNU parallel（cf. Tange, 2018）を利用して並列処理により高速化することも試みている．

与えている．なお，これらのデータベースの構築に関する詳細については，付録 B を参照されたい．

V　抽出システムにおけるインターフェースデザインの改良と財務データとそのダウンロード法の拡充

課題（P4），（P5）に対して，KGUSBADES によるサービスが2010年に開始されて以来，2012年にサービスの種類を増加させることに伴って，Web インターフェースのデザインを改良したが，今回の再構築では，システム名称の変更と抽出できるデータの種類をさらに拡充する予定があるため，それに伴って図 3 のように Web インターフェースを改良することを検討した．なお，このシステムに関する詳細は，付録 C を参照されたい．

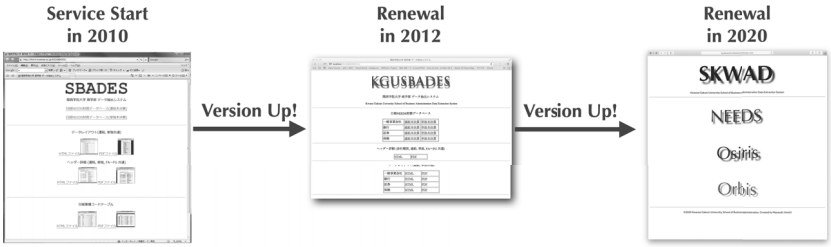


図 3 ：Web インターフェースの改良

本システムを公開・運用・メンテナンスする中で，筆者が Web インターフェースをデザインする際，重視してきたコンセプトは「シンプル」という一点のみである．本システムは，不特定多数に対して公開するわけでなく，学内限定であり，かつその利用に関しては，教育・研究に利用されるものであることから，極力余計な情報を提供することなく，データソース名をアイコン化することによってシンプルなものとした．今回リニューアルが予定されているシステムのトップページは，（システムのロゴアインコン以外では）NEEDS，Osiris，Orbis という財務データの提供元が命名したデータベース名を表すアイコンが並んでいるだけのものであり，ユーザはファーストアク

セス時には戸惑うかもしれないが，簡単な説明を与えれば，それ以降は混乱はないものと思われる．

これらのアイコンは，以下のようなデータを抽出するためのものである：

NEEDS: 日経メディアマーケティング株式会社[10] から販売されている日本における一般事業会社（1.6万社超）の財務データ（NEEDS 企業財務データ）

Osiris: ビューロー・ヴァン・ダイク（Bureau van Dijk: BvD）社[11] から販売されている世界の全上場会社（9 万社超）の財務データ

Orbis: BvD 社から販売されている世界の（財務データが収集されている）全会社（2,600万社超）の財務データ

本稿では，これらの選択肢のうち，NEEDS に関するデータ抽出に関する仕様を説明し，Osiris と Orbis については別の機会に譲る[12]．

NEEDS 企業財務データ（一般事業会社）を抽出するためには，トップページから NEEDS ロゴ（アイコン NEEDS）をクリックし，移動したページのリンク 日経 NEEDS 財務データベース（2019年版） をクリック後，リンク 連結本決算 またはリンク 単独本決算 のどちらかを選択することによって抽出システムのページに移動することができる（図 4）．

ここでは，リンク 連結本決算 を選択した場合，すなわち「NEEDS 企業財務データ抽出システム（一般事業会社：2019年版，連結本決算）」の画面（図 5）を用いて説明する．まず，SQL 問合せ（SQL query, SQL については，付録 A・3 参照）のスクリプトを スクリプト入力ボックス に入力し，Submit ボタンをクリックすることによって，サーバに命令が送信され，結

10）https://www.nikkeimm.co.jp/
11）https://www.bvdinfo.com/
12）Osiris，Orbis のデータ抽出については，主要な財務指標に限定したサービスを行う予定である．特に，Orbis については規模の問題から，一部の企業に限定する予定である．

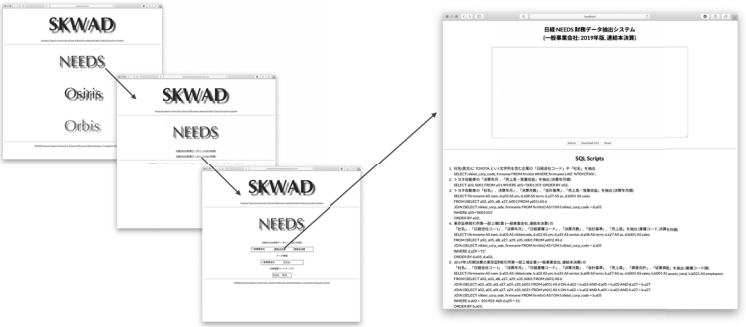


図 4 ：NEEDS 企業財務データ抽出システム（一般事業会社：2019年版，連結本決算）へのリンク

果が HTML 形式で返信される．次に，SQL 問合せのスクリプトを スクリプト入力ボックス に入力し，Download CSV ボタンをクリックすることによって，サーバに送信された命令の結果を CSV 形式でダウンロードすることができる．また，Reset ボタンをクリックすると スクリプト入力ボックス内 のスクリプトがクリアされる．なお，スクリプト入力ボックス の大きさはボックスの右隅にあるリサイズアイコン（//）を使って調整することができる．

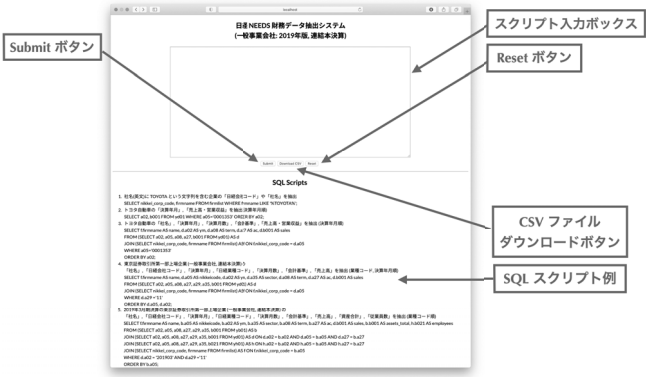


図 5 ：NEEDS 企業財務データ抽出システム（一般事業会社：2019年版，連結本決算）のページ

なお，SQL のサンプルスクリプトも用意されている（図 5 の下部を参照）ので，コピー・アンド・ペーストすることによって，手軽にデータ抽出を試すことができる．

では，以下に SQL 問合せの例を幾つか取り上げる．

1. トヨタ自動車の日経会社コードの検索

最初の例として与えられているトヨタ自動車の日経会社コードを検索するための操作を図 6 に与える．

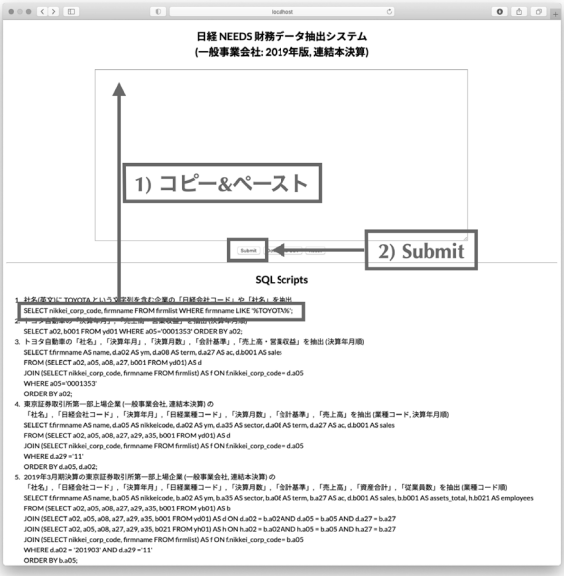


図 6 ：トヨタ自動車の日経会社コードの検索

この抽出に利用された SQL 問合せ（スクリプト）をスクリプト 1 に与える．テーブルと列名の意味については，付録 B か日本経済新聞社（2019）を参照されたい．

スクリプト１：社名に 'TOYOTA' という文字列を含む企業の日経会社コードと社名を抽出

```
1 SELECT nikkei_corp_code, firmname
2       FROM firmlist
3             WHERE firmname LIKE '%TOYOTA%';
```

この SQL 問合せを以下に説明する．

1 行目：SELECT 句で日経会社コード（nikkei_corp_code）と社名（firmname）の列を指定する．

2 行目：FROM 句で日経会社コード（nikkei_corp_code）や社名（firmname）が納められているテーブル firmlist を指定する．

3 行目：WHERE 句で条件として英文表記の企業名（firmname）の列において TOYOTA という文字列を含むような（LIKE）行に限定する．

ここで，% は任意の文字列を表し，'（シングルクォート）は条件の文字列を「包む」ための記号であり，「クォート処理」と呼ばれることがある．

2. トヨタ自動車の売上高の抽出

次に，実際の企業の財務データの抽出例として，トヨタ自動車の売上高の推移を表わす時系列プロットを与える．実際の操作手順は以下のようなものである（図 7）：

1. SQL のサンプルスクリプトの 2 番目を スクリプト入力ボックス にコピー・アンド・ペースト
2. Submit ボタンをクリック
3. 得られた抽出結果を，全て選択し，Microsoft Excel（以下 Excel と略）にコピー・アンド・ペースト
4. セルを適切に選択し，時系列プロット

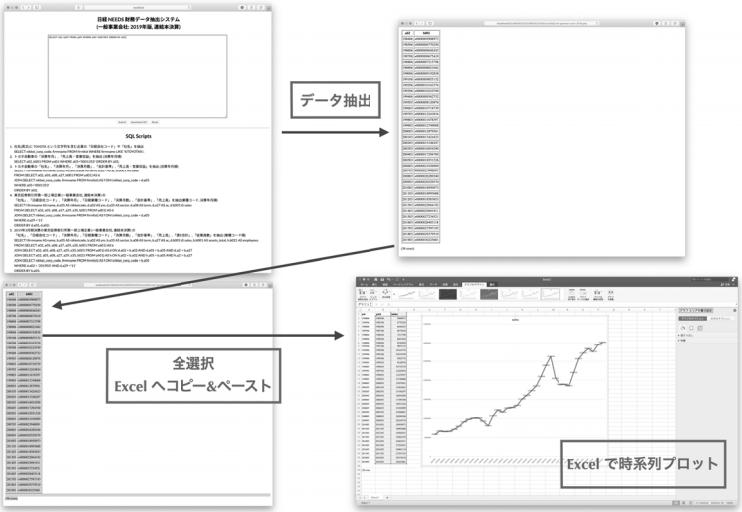


図 7 ：トヨタ自動車の売上高の推移（時系列プロット）の作成工程

この抽出に利用された SQL 問合せ（スクリプト）は以下のようなものである：

スクリプト 2 ：トヨタ自動車の「決算年月」,「売上高・営業収益」の抽出（決算年月順）

```
1 SELECT a02, b001
2       FROM yd01
3             WHERE a05='0001353'
4                   ORDER BY a02;
```

この SQL 問合せを以下に説明する.

1 行目：SELECT 句で決算年月（a02）と売上高（b001）の列を指定する.

2 行目：FROM 句で決算年月（a02）と売上高（b001）がおさめられているテーブル yd01 を指定する.

3 行目：WHERE 句で列 a05（日経会社コード）においてトヨタ自動車の日経会社コード 0001353 を含む行（a05='0001353'）を限定する.

4 行目：ORDER BY 句で決算年月（a02）の順に並べ替える.

3. 2019年3月期決算の東京証券取引所第一部上場企業（一般事業会社，連結本決算）の財務データの抽出

さらに複雑なSQL問合せの例として，2019年3月期決算の東京証券取引所第一部上場企業（一般事業会社，連結本決算）の売上高や資産合計，従業員数を抽出する例を考える．データの抽出は，トヨタ自動車の売上高を抽出する手順と本質的には変わらない（図8参照）．

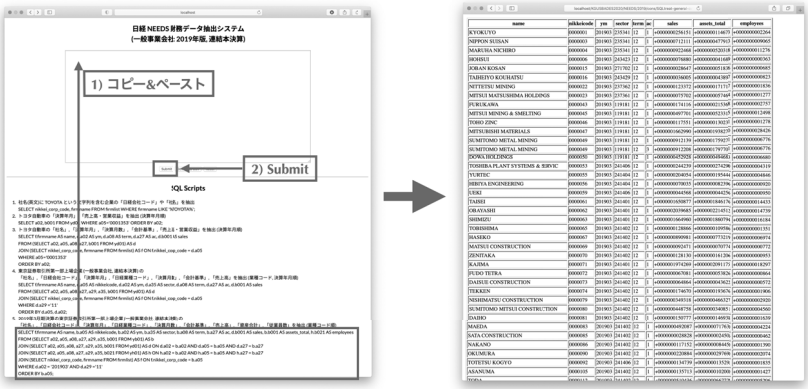


図8：2019年3月期決算の東京証券取引所第一部上場企業（一般事業会社，連結本決算）の売上高，資産合計，従業員数等の抽出

この抽出に利用されたSQL問合せ（スクリプト）は以下のようなものである：

スクリプト3：2019年3月期決算の東証1部上場企業（連結本決算）の「売上高」，「資産合計」，「従業員数」等の抽出

```
SELECT
      f.firmname AS name,
      b.a05 AS nikkeicode,
      b.a02 AS ym,
      b.a35 AS sector,
      b.a08 AS term,
      b.a27 AS ac,
      d.b001 AS sales,
      b.b001 AS assets_total,
```

```
10        h.b021 AS employees
11        FROM (SELECT a02, a05, a08, a27, a29, a35, b001 FROM yb01) AS b
12        JOIN (SELECT a02, a05, a08, a27, a29, a35, b001 FROM yd01) AS d
13              ON d.a02 = b.a02 AND d.a05 = b.a05 AND d.a27 = b.a27
14        JOIN (SELECT a02, a05, a08, a27, a29, a35, b021 FROM yh01) AS h
15              ON h.a02 = b.a02 AND h.a05 = b.a05 AND h.a27 = b.a27
16        JOIN (SELECT nikkei_corp_code, firmname FROM firmlist) AS f
17              ON f.nikkei_corp_code = b.a05
18              WHERE d.a02 = '201903' AND d.a29='11'
19                    ORDER BY b.a05;
```

この SQL 問合せの説明を以下に与える．なお，説明の都合上，行に関して順不同となっている箇所がある．

11行目：まず，括弧内（...）で SELECT 文によってデータを抽出している．ここでは，FROM 句に貸借対照表のテーブル yb01 を指定し，SELECT 句でテーブル yb01 のヘッダーパートから，決算年月（a02），日経会社コード（a05），決算月数（a08），会計基準（a27），上場場部（a29），日経業種コード（a35）を抽出し，同じテーブル yb01 のデータパートから，資産合計（b001）の列を抽出し，AS 句でその結果をテーブル名 b と定義している．次に，そのテーブル b を括弧外の FROM 句で指定する．

12行目～13行目：まず，括弧内（...）で SELECT 文によってデータを抽出している．ここでは，FROM 句に損益計算書のテーブル yb01 を指定し，SELECT 句でヘッダーパートから，テーブル b と同じ列を抽出し，さらにデータパートから，売上高（b001）の列を抽出した後，AS 句でそれらの結果をテーブル名 d として定義している．次に，ON 句で決算年月（a02），日経会社コード（a05），会計基準（a27）をキーとして指定（ON d.a02 = b.a02 AND d.a05 = b.a05 AND d.a27 = b.a27）し，JOIN 句でテーブル b へ結合している．

14行目～15行目：まず，括弧内（...）で SELECT 文によってデータを抽出している．ここでは，FROM 句に「その他・明細情報等」のテーブル yh01 を指定し，SELECT 句でヘッダーパートから，テーブル b

と同じ列を抽出し，さらにデータパートから，（期末）従業員数（b021）を抽出した後，AS 句でそれらの結果をテーブル名 h として定義している．次に，ON 句で決算年月（a02），日経会社コード（a05），会計基準（a27）をキーとして指定（ON d.a02 = b.a02 AND d.a05 = b.a05 AND d.a27 = b.a27）し，JOIN 句でテーブル b へ結合している．

16行目～17行目：まず，括弧内（...）で SELECT 文によってデータを抽出している．ここでは，FROM 句に収録会社情報のテーブル firmlist を指定し，SELECT 句で日経会社コード（nikkei_corp_code）と社名（firmname）を抽出し，AS 句でその結果をテーブル名 f として定義している．次に，ON 句で日経会社コードをキーとして指定（ON f.nikkei_corp_code = b.a05）し，JOIN 句でテーブル b へ結合している．

18行目：WHERE 句で決算年月が2019年 3 月（d.a02 = ′201903′）と上場場部が東京証券取引所第一部（d.a29 =′11′）の行を抽出する条件を与えている．

19行目：ORDER BY 句で日経会社コード（b.a05）の昇順で並べ替えている．

１行目～10行目：AS 句を伴って，SELECT 句で社名（f.firmname）を name，日経会社コード（b.a05）を nikkeicode，決算年月（b.a02）を ym，日経業種コード（b.a35）を sector，決算月数（b.a08）を term，会計基準（b.a27）を ac，売上高（d.b001）を sales，資産合計（b.b001）を assets_total，従業員数（h.b021）を employees として抽出している．

トヨタ自動車の売上高の例と同様に抽出結果を Excel へコピー・アンド・ペーストし，可視化（売上高と資産合計の散布図を作成）する工程を図 9 に与える．

図 9 ：2019年 3 月期決算の東証一部上場企業（連結本決算）の売上高（x-軸）と資産合計（y-軸）の散布図の作成

従来の財務データ抽出システム（KGUSBADES）では，ここで利用しているような，Web ブラウザから Excel へコピー・アンド・ペーストを行うことによってデータを可視化・分析することを前提としてきた[13]．トヨタ自動車の売上高を多年度にわたって抽出する例では抽出されたデータは36行であったので，この手順は手軽さということでは適していると思われる．しかしながら，このような仕様は以下のような意味で限界があるように思われる．

（EP1）スクリプト 3 で与えられる抽出例では，結果が1000行を超える（1346行）ため，「コピー・アンド・ペースト」を「手作業」で行う段階で操作上の誤りをおかす可能性がある．

（EP 2）Excel などのグラフィカル・ユーザ・インターフェース（Graphical User Interface: GUI）をベースとするソフトウェアで可視化を行った結果は，手法の種類が限定されることやその再現性が乏しい．

問題（EP1）は，本稿の冒頭で述べた課題（P6）「ダウンロード法の拡充」によってある程度対処することができる．今回のシステム再構築では，

13）ここで，Web ブラウザから Excel へコピー・アンド・ペーストを行う場合でも，結果を CSV ファイルに保存し，他のデータ解析ソフトウェアに読み込んで可視化・分析することが可能である．筆者が担当する講義や演習では，高度な可視化・分析を行う際には，Excel 以外の専門的なソフトウェアで実施することを推奨してきた．

Download CSV ボタンを用意することによって，CSV ファイルとしてデータをダウンロードするための機能を追加した．例えば，2019年 3 月期決算の東京証券取引所第一部上場企業（一般事業会社，連結本決算）の売上高，資産合計，従業員数等の抽出した結果を CSV ファイルとしてダウンロードするためには，SQL 問合せのスクリプトを スクリプト入力ボックス にコピー・アンド・ペースト後，Download CSV ボタンをクリックすればよい（図10参照）．

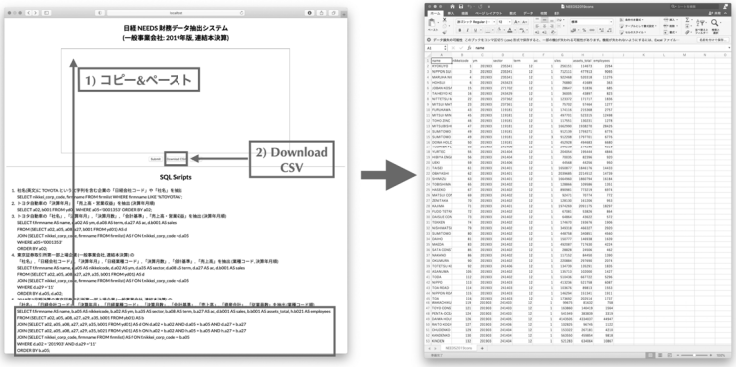


図10：2019年 3 月期決算の東京証券取引所第一部上場企業（一般事業会社，連結本決算）の売上高，資産合計，従業員数等の抽出結果を CSV ファイル NEEDS2019cons.csv としてダウンロード

このようにダウンロードされた CSV ファイル NEEDS2019cons.csv は，Excel のみならず R[14] などのデータ解析環境に読み込んで可視化・分析・解析することも可能である．例えば，適当な場所（作業ディレクトリ）に保存された CSV ファイル NEEDS2019cons.csv を R に読み込むには，read.csv 関数を利用して以下のように入力すればよい[15]．

14）https://www.r-project.org/
15）R を起動後，作業ディレクトリを適切に設定する必要がある．詳細は地道（2018-c）等を参照されたい．

Rへのデータの読み込み

```
> x <- read.csv("NEEDS2019cons.csv")
```

ここで > はRのプロンプトである．このように読み込まれたオブジェクトの先頭 6 行は関数 head を使って以下のように表示できる．

読み込んだデータオブジェクト x

```
> head(x)
                    name nikkeicode     ym sector term ac  sales assets_total employees
1 KYOKUYO                        1 201903 235341   12  1 256151       114673      2264
2 NIPPON SUISAN                  3 201903 235341   12  1 712111       477913      9065
3 MARUHA NICHIRO                 4 201903 235341   12  1 922468       520318     11276
4 HOHSUI                         6 201903 243423   12  1  76880        41689       363
5 JOBAN KOSAN                   15 201903 271702   12  1  28647        51836       685
6 TAIHEIYO KOUHATSU             16 201903 243429   12  1  36005        43897       823
```

ここで，変数名（列名）は以下のようなものである：

name：企業名

nikkeicode：日経会社コード

ym：決算年月

sector：日経業種コード

term：決算月数

ac：会計基準（1：日本基準，2：米国会計基準，3：国際会計基準）

sales：売上高（単位：百万円）

assets_total：資産合計（単位：百万円）

employees：従業員数（単位：人）

このデータオブジェクトの対散布図をプロットすることを考える．そのためには，以下のようにまずオブジェクトを変換する必要がある．

データオブジェクトの変換

```
> library(dplyr)
> y <- x %>%
+   mutate(sales = na_if(sales, "-9999999999999"),
+          assets_total = na_if(assets_total, "-9999999999999"),
+          employees = na_if(employees, "-9999999999999"),
+          sector1 = as.factor(substring(sector,1,1)))
```

このRスクリプトでは，データ操作（data manipulation）を行うためのRパッケージdplyrを読み込んでおり，このパッケージに付属のmutate関数を利用して，売上高（sales），資産合計（assets_totala），従業員数（employees）に含まれる欠測値（-9999999999999[16]）を，関数na_ifを利用して，欠測値（NA）として変換し，変数を再定義している．また，日経業種コード（sector）の1文字目が大分類（1：製造業，2：非製造業）を表すことから，関数substringを利用して切り出し，さらにそれを関数as.facorを利用して，因子型に変換したものを新しい列sector1として追加している．

以上の準備のもとで，以下のようなスクリプトを実行することによって対散布図をプロットする．

Rによる対散布図のプロット：ggpair関数を利用した場合

```
> library(GGally)
> y %>% select(sales, assets_total, employees, sector1) %>%
+                 ggpairs(
+                 mapping = aes(color = sector1),
+                 upper = list(continuous = wrap("points", size = 0.5, alpha = 0.5)),
+                 lower = list(continuous = wrap("cor", size = 3))
+                 ) +
+                 theme(
+                 axis.text= element_text(size = 5),
+                 axis.title = element_text(size = 3)
+                 )
```

このRスクリプトでは，関数libraryをつかってGGallyパッケージを呼び出し，そのパッケージに付属するggpair関数を利用して対散布図を描いている．パイプ演算子%>%を利用して，パイプラインを構成することによって，データのオブジェクトyの列を選択するためにselectに引き渡し，さらに，対散布図を描くために関数ggpairsに引き渡している．なお，日経業種分類（大分類sector1）で色分けするために，審美的属性（aesthetic attributes）を与えるための関数aesを利用して引数mappingに情報

16）NEEDS企業財務データでは，欠測値は-9999999999999で表される．

(aes(color = sector1)) を与えており，引数 upper と lower に対散布図の上三角ブロックと下三角ブロックに，それぞれ，散布図の点 ("point") と相関係数の値 ("cor") を与えることを点と文字の大きさとともに指定している．それ以外の指定は，軸などに利用される文字の大きさを調整するものである．詳細は，ggpairs のヘルプを参照されたい．

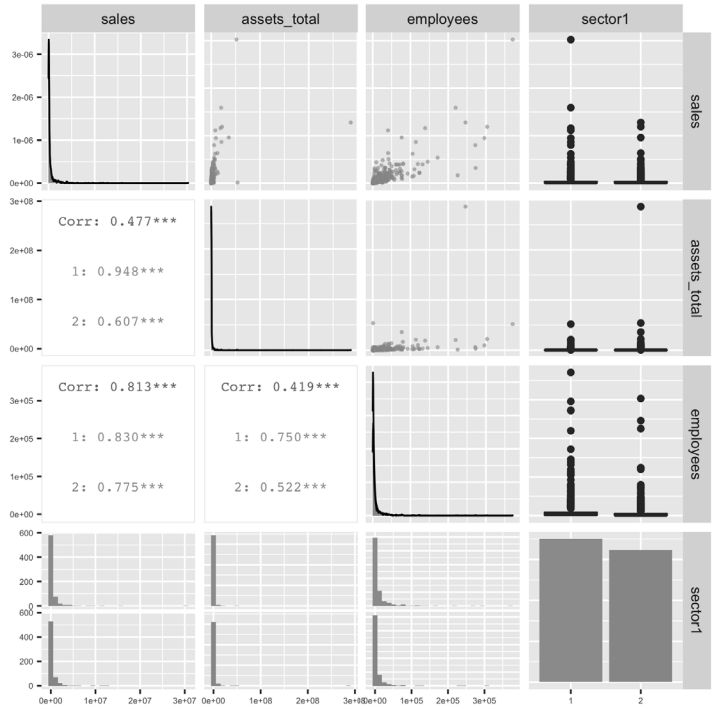


図11：対角ブロックには売上高（sales），資産合計（assets_total），従業員数（employees）に関する推定された密度関数と，日経産業分類（sector1：大分類）の頻度が描かれている．また，上対角ブロックには，売上高（sales），資産合計（assets_total），従業員数（employees）の 2 つの組合せに関する散布図と最終列には，それぞれのデータのボックスプロット（1：製造業，2：非製造業）が描かれている．さらに，下対角ブロックには，売上高（sales），資産合計（assets_total），従業員数（employees）の 2 つの組合せに関する相関係数が与えられており，最終行には，それぞれのデータ頻度のバーチャート（1：製造業，2：非製造業）が描かれている．

対散布図（図11）から，業種によらず売上高（sales），資産合計（assets_total），従業員数（employees）の全てが極端に右に歪んだものであることがわかる．また，これらのペアの散布図も原点付近に密集しており，2次元の意味で歪んだものであることがわかる．さらに，業種（1：製造業，2：非製造業）によって，相関の強弱に違いがあることがわかる．例えば，売上高（sales）と資産合計（assets_total）の相関は，製造業（0.948）と非製造業（0.607）で極端に異なっていることがわかる．このように，Rを利用して可視化することによって，ここで扱っている財務データの特性を詳細に捉えることができる．また，この結果はデータとSQL，Rスクリプト（コード）を適切に管理することによって，簡単に再現できることも利点といえる[17]．以上のことから，問題（EP2）に対しては，ここで述べた方法が解決法の一つとなろう（図12も参照）．

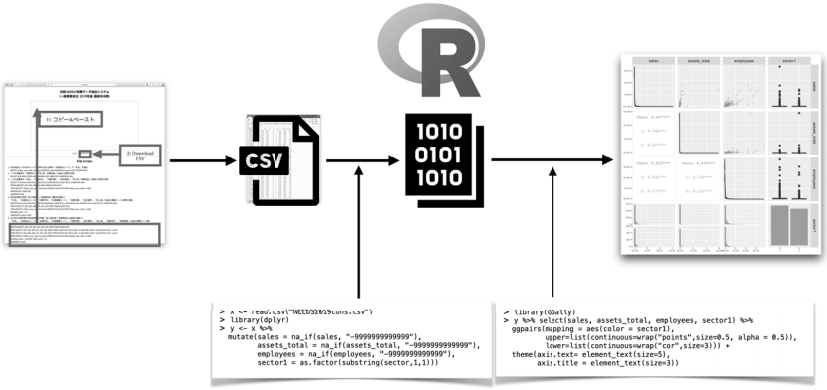


図12：CSVファイルとしてダウンロードされたデータファイルをRを利用して可視化する工程

17）本稿は，Sweave（https://stat.ethz.ch/R-manual/R-devel/library/utils/doc/Sweave.pdf）を利用することによって，LaTeXにRのコードを埋め込み，自動実行することによって動的に文書を生成する方法で作成している．

VI おわりに

本稿では，2010年から運用してきた学内向け財務データ抽出システムの様々な課題に対する対応策を検討することによって再構築を試みた．財務データ Osiris と Orbis にもとづくデータ抽出システムについては詳しく述べることができなかったが，別の機会に譲ることとする．なお，これらのデータに対する前処理やデータラングリングについての詳細は，地道（2018-a, b, 2020-a, b）を参照されたい[18]．

また，今回再構築したシステムは，安定性を重視したため，Apache Web Server，MySQL，PostgreSQL，PHP などの「堅実」なシステムの組み合わせを利用しているが，これらのシステムに代わるものは，日々新しいものが生み出されており，常に新しい情報をキャッチアップすることは重要であろう．

前回のシステム開発から，10年の歳月を経て，このシステムを利用した研究成果もいくつかでている（cf. 地道, 2014, Jimichi and Maeda, 2014, 柳ら, 2018-a, b, 2019, Shih et al., 2020）[19]．今回のシステム再構築によって，さらなる研究成果が生み出されることが期待される．

（筆者は関西学院大学商学部教授）

参考文献

［1］ Codd, E. F.（1970）A relational model of data for large shared data banks, *Communications of the ACM,* Vol. 13, pp. 377-387.

［2］ Janssens, J.（2014）*Data Science at the Command Line,* O'Reilly Media.（太田満久,

18）現時点で，地道（2018-a, b, 2020-a, b）は全て関西学院大学のリポジトリサーバ（https://kwansei.repo.nii.ac.jp/）から PDF ファイルがダウンロード可能である．

19）筆者が2017年から担当している講義（商学部開講科目「ビジネスデータ分析 I，II」）では，このシステムを利用し，構造化参照言語（Stractured Query Language: SQL）を用いた財務データの抽出から，前処理・可視化・統計モデリングを実行することを実施している．柳ら（2018-a, b, 2019）の研究成果は，この講義の課題レポートに端を発するものである．

下田倫大，増田泰彦監訳，長尾高弘訳（2015）『コマンドラインではじめるデータサイエンス：分析プロセスを自在に進めるテクニック』，オライリー・ジャパン.）

［3］ 地道正行（2010-a）『日経 NEEDS 財務データにもとづくデータベースサーバの構築』，商学論究，第57巻，第 4 号，pp. 23-80，関西学院大学商学研究会.

［4］ 地道正行（2010-b）『財務データベースサーバの構築』，関西学院大学リポジトリ，http://hdl.handle.net/10236/6013，ISBN: 9784990553005

［5］ 地道正行（2012）『金融関連会社に関する日経 NEEDS 財務データにもとづくデータベースの構築』，商学論究，第59巻，第 3 号，pp. 35-70，関西学院大学商学研究会.

［6］ 地道正行（2014）『R を利用した財務データの可視化と統計モデリング－探索的データ解析の視点から－』，商学論究，第61巻，第 3 号， pp. 241-295，関西学院大学商学研究会.

［7］ Jimichi, M. and S. Maeda（2014）Visualization and Statistical Modeling of Financial Data with R, Poster at *The R User Conference 2014,* University of California, Los Angeles, USA, July 1st, 2014.

［8］ 地道正行（2018-a）『探索的財務ビッグデータ解析－前処理，データラングリング，再現可能性－』，商学論究，第66巻，第 1 号，pp. 1-32，2018年 9 月，関西学院大学商学研究会.

［9］ 地道正行（2018-b）『探索的財務ビッグデータ解析－データ可視化，統計モデリング，モデル選択，モデル評価，動的文書生成，再現可能研究－』，商学論究，第66巻，第 2 号， pp. 1-41，2018年12月，関西学院大学商学研究会.

［10］ 地道正行（2018-c）『データサイエンスの基礎：R による統計学独習』，裳華房.

［11］ 地道正行（2020-a）『探索的財務ビッグデータ解析－前処理の並列化－』，商学論究，第67巻，第 3 号，pp. 1-19，関西学院大学商学研究会.

［12］ 地道正行（2020-b）『探索的財務ビッグデータ解析－PG-Strom によるデータラングリングの並列化－』，商学論究，第68巻，第 1 号，pp. 1-34，関西学院大学商学研究会.

［13］ 増永良文（2017）『リレーショナルデータベース入門－データモデル・SQL・管理システム・NoSQL－』，サイエンス社.

［14］ 日本経済新聞社（2018）『NIKKEI NEEDS 一般事業会社企業財務データ項目定義書2018年12月17日版』，日本経済新聞社.

［15］ 日本経済新聞社（2019）『NIKKEI NEEDS 一般事業会社企業財務データ2019年10月 1 日版』，日本経済新聞社.

［16］ 西沢夢路（2017）『基礎からの MySQL 第 3 版』，SB クリエイティブ.

［17］ Shih, J, T. Lin, M. Jimichi, and T. Emura（2020）Robust ridge M-estimators with pretest and Stein-rule shrinkage for an intercept term, *Japanese Journal of Statistics and Data Science,* DOI：10.1007/s42081-020-00089-6.

［18］ 鈴木啓修（2012）『PostgreSQL 全機能バイブル』，技術評論社.

［19］ Taddy, M.（2019）*Business Data Science: Combining Machine Learning and Economics*

to Optimize, Automate, and Accelerate Business Decisions, McGraw-Hill.（上杉隼人，井上毅郎共訳（2020）『ビジネスデータサイエンスの教科書』，すばる舎.）

［20］ Tange, Ole,（2018）*GNU Parallel 2018,* ISBN: 9781387509881, DOI: 10.5281/zenodo.1146014, URL: https://doi.org/10.5281/zenodo.1146014, Mar, 2018.

［21］ Wickham, H. and G. Grolemund（2016）*R for Data Science,* O'Reilly.

［22］ 柳 麻衣，阪 智香，地道正行（2018-a）『配当金支払金額の探索的データ解析』，国際数理科学協会2018年度年会，「統計的推測と統計ファイナンス」分科会研究集会，関西学院大学梅田キャンパス1402号教室，2018年 8 月25日（土）.

［23］ 柳 麻衣，阪 智香，地道正行（2018-b）『配当金支払金額の探索的データ解析』，2018年度統計関連学会連合大会，中央大学後楽園キャンパス，2018年 9 月12日（水）.（発表要旨，http://www.jfssa.jp/taikai/2018/table/program_detail/pdf/51-100/J10096.pdf）

［24］ 柳 麻衣，阪 智香，地道正行（2019）『配当金の探索的データ解析』，第13回日本統計学会春季集会，ポスター発表，日本大学経済学部本館，2019年 3 月10日（日）.

謝辞

本研究の一部は以下の助成を得ている.

科研費科学研究費基盤研究 C：「グラフィカル・データ・アナリシスによる格差研究と社会環境会計による解決方法の提案」（2016年～2019年），課題番号：16K04022

科研費科学研究費基盤研究 C：「共有価値創造（CSV）のための社会環境会計の構築」（2019年～2021年），課題番号：19K02006

2019年度学際大規模情報基盤共同利用・共同研究拠点（JHPCN）課題：「財務ビッグデータの可視化と統計モデリング」，課題番号：jh191002-NWJ

2020年度学際大規模情報基盤共同利用・共同研究拠点（JHPCN）課題：「財務ビッグデータの可視化と統計モデリング」，課題番号：jh201003-NWJ

関西学院大学図書館図書費 B，研究設備費（III），個人研究費

また，日経メディアマーケティング株式会社佐藤健吾氏，BvD 社増田歩氏，関西学院大学商学部阪智香教授，関西学院大学商学部研究資料室高瀬忍氏，関西学院大学図書館藤澤快氏には様々なご足労を賜った．ここに感謝の意を表する．本稿では，以下のサイトから提供されているアイコンを利用させていただいた．重ねて感謝の意を表する．
https://www.silhouette-illust.com/

付録 A　リレーショナル・データベースに関する基礎知識

ここでは，地道（2010）を参考に，データベースを理解する上で最も基本的な用語・知識を与える．なお，詳しくは増永（2017）などを参照されたい．

A. 1　リレーショナル・データ・モデル

リレーショナル・データベースのもとになる概念であるリレーショナル・データ・モデルは Codd（1970）によるものである．ここでは，それに関連する用語をまとめる．

ドメイン（domain）：集合のこと．記号として $D_1, \ldots, D_n$ で表される．

直積（Cartesian product）：

$$D_1 \times \cdots \times D_n := \{(d_1, \ldots, d_n) \mid d_i \in D_i,\ i = 1, \ldots, n\}$$

をドメイン $D_1, \ldots, D_n$ の直積（集合）という．

タプル（tuple），レコード（record）：ドメインの直積集合 $D_1 \times \cdots \times D_n$ の要素：

$$t := (d_1, \ldots, d_n) \in D_1 \times \cdots \times D_n$$

のこと．

リレーション（relation）：ドメインの直積集合 $D_1 \times \cdots \times D_n$ の任意の有限部分集合：

$$R \subset D_1 \times \cdots \times D_n$$

のこと．すなわち，タプルを要素とする集合を表す．なお，リレーションが定義されているドメインの個数を次数（degree）という．

リレーショナル・データベース（Relational DataBase; RDB）：リレーショナル・データ・モデルにもとづいて設計，開発されるデータベース

リレーショナル・データベース管理システム（Relational DataBase Management System; RDBMS）：リレーショナル・データベースを管理するためのソフトウェア

属性（attribute）：タプルがあらわしている対象のもつ属性．属性名を $A_1, \ldots,$

A_n で表す.

ドメイン関数：属性 A_i からドメイン D_i への写像：

$$\mathrm{dom} : A_i \to D_i,\ i=1, \ldots, n$$

をドメイン関数（domain function）という.

リレーションスキーマ（relation schema）：リレーションの時間的に不変な性質

インスタンス（instance）：時間とともに変化するリレーションそのもの

シンプル（simple）：ドメインが他のドメインの入れ子（nest）や巾集合（power set）等であらわすことができないこと.

第 1 正規形（the first Normal Form; 1NF）：リレーションを定義するすべてのドメインがシンプルであること．通常のリレーションには第 1 正規形であることが仮定されている.

注意 A. 1. リレーションは 2 次元の単純なテーブル（table）と考えてもよく[20]，タプルはその行（row），テーブルの列（column）は属性に対応し，ドメインは属性の取り得る値全体の集合といってもよい.

A. 2 リレーショナル・データベース管理システム

リレーショナル・データベース（RDB）を管理するシステムがリレーショナル・データベース管理システム（RDBMS）である．オープンソースとしては，MySQL，PostgreSQL，SQLite 等が存在し，さまざまな OS 上で稼働している．リレーショナル・データベースは，テーブル（表，リレーション）から構成されており，テーブルにおける縦の並びをカラム（列，フィールド），横の並びをレコード（行，タプル）と呼ぶ（図13も参照）.

20）ここでは，「ぶち抜き」や「入れ子」のないことを前提としている

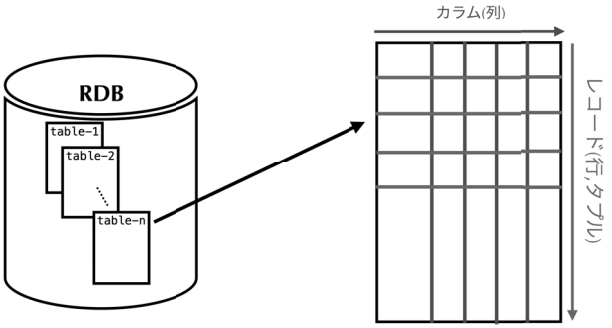


図13：リレーショナル・データベース（RDB）の構造とテーブル

A. 3　構造化照会言語

RDBMS とのインターフェース言語として構造化照会言語（Structured Query Language; SQL）が国際標準として利用されている（図14参照）．この言語は，米国国家規格協会（American National Standards Institute; ANSI）と国際標準化機構（International Organization for Standardization; ISO）によって制定されたものである． SQL92 が ISO/IEC9075 と ANSI X3.135-1992 で制定されており，日本ではその邦訳が日本工業規格（Japanese Industrial Standard; JIS）の JIS X3005 として制定されている．

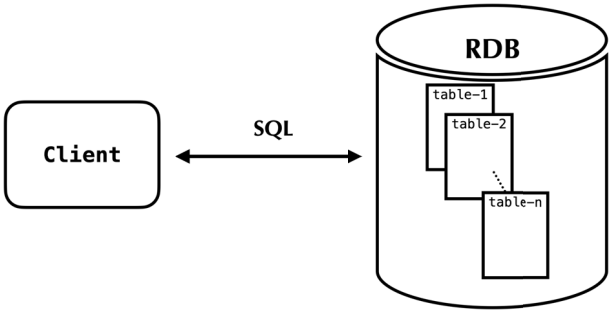


図14：SQL の役割

リレーショナル・データ・モデルにおける用語と SQL における用語には

以下の対応がある：

リレーショナル・データ・モデル	SQL
リレーション（relation）	テーブル（table）
属性（attribute）	カラム，列（column）
タプル（tuple）	ロウ，行（row），レコード（record）

SQL によってテーブルからカラムを抽出するためは SELECT 文（ステートメント）を利用する．基本的な構文は以下のようなものである：

SELECT 文の基本構文

```
SELECT カラム名1, カラム名2,…
        FROM テーブル名
        JOIN テーブル名 ON 条件
                WHERE 条件 (LIKE 条件)
                                ORDER BY カラム名;
```

ここで，SELECT から FROM の間でカラム名を指定（SELECT 句）し， FROM でテーブル名を選択するが，これらは必須である．また，それ以外の JOIN（テーブルの結合），WHERE（条件指定），ORDER BY（並べ替え）は，オプショナルである．なお，SELECT 句にはワイルドカードとしてアスタリスク（*）を与えことができ，そのテーブルの全てのカラム（列）が選択される．

付録 B　NEEDS 企業財務データ（一般事業会社）2019年版にもとづくデータベースの構築

B. 1　データ仕様

ここでは，日経メディアマーケティング社が販売している「NEEDS 企業財務データ（一般事業会社）」のファイルの2019年版に関する情報を与える（表 2 参照）[21]．

21）本稿では2018年版のデータも利用しているが，構造は2019年版と同様である．

表 2 ：NEEDS 企業財務データ（一般事業会社）：2019年版

ファイル名	ファイルサイズ（Byte）	説明
HFSCNM010.zip	1300591	収録会社情報ファイル（圧縮済み）
HFSIDA920.1960.zip	9258983	1960 年代のデータファイル（圧縮済）
HFSIDA920.1970.zip	19372613	1970 年代のデータファイル（圧縮済）
HFSIDA920.1980.zip	28242996	1980 年代のデータファイル（圧縮済）
HFSIDA920.1990.zip	44883899	1990 年代のデータファイル（圧縮済）
HFSIDA920.2000.1.zip	72354532	2000 年代のデータファイル 1（圧縮済）
HFSIDA920.2000.2.zip	16457461	2000 年代のデータファイル 2（圧縮済）
HFSIDA920.2010.1.zip	78595310	2010 年代のデータファイル 1（圧縮済）
HFSIDA920.2010.2.zip	7843009	2010 年代のデータファイル 2（圧縮済）

収録会社情報のファイル HFSCNM010.zip とデータファイル HFSIDA920.1960.zip〜HFSIDA920.2010.2.zip で構成されており，全て ZIP 形式で圧縮されている．

収録会社情報のファイル HFSCNM010.zip を展開後，文字コードを UTF-8 に変換し，先頭の 1 行を表示した結果をリスト 4 に与える．

スクリプト 4 ：収録会社情報のファイル HFSCNM010.zip の先頭 1 行

```
% 7z e -so HFSCNM010.zip | nkf -u | head -n 1
CN0000000011301 9851662019999          KYOKUYO                          極洋
                キヨクヨウ                        東京都港区赤坂３－３－５住友
    生命山王ビル                    107005203-5545-0701
                                2353411010401033225
```

展開後のファイルは，1 行300バイト（Byte）であり，フィールドとフィールドを分ける区切り文字[22] は無く，いわゆる，固定長フォーマット（fixed length format）のファイルである．ファイルに含まれる項目の仕様は，表 3 を参照されたい．なお，このファイルをデータベースのテーブルとして利用

22）区切り文字（separator）とは，テキストデータ中で複数の要素を並べて記述する際に，要素の区切りを表す記号や特殊な文字（の並び）のことである．デリミタ，セパレータ，分離記号，分離文字などとも呼ばれる．列挙された項目の区切りを表すものと，範囲の始まりと終わりを表すものがある（IT 用語辞典 e-Words http://e-words.jp/参照）．

するためには，表3に与えられているように，項目の「位置」と「長さ」の情報をもとに適切に区切り文字を入れたり，「カラム名」を設定するなどの処理が必要となる．

表3：収録会社情報のファイル HFSCNM010 の仕様

項番	項目名	説明	カラム名	位置	長さ
1	レコード種別	CN00：固定	record_type	1	4
2	日経会社コード	日経が定める会社コード	nikkei_corp_code	5	7
3	株式コード		stock_code	12	4
4	予備		etc1	16	1
5	予備		etc2	17	9
6	金融機関コード	証券コード協議会が定める4桁の会社コード	finance_code	26	4
7	予備		etc3	30	11
8	英文略称	英文による会社名の略称（但し英文商号のない時はローマ字）	firmname	41	30
9	漢字略称	会社名の漢字略称	firmname_jp	71	30
10	カナ社名	カタカナによる会社名の略称（漢字モードのカタカナ表記）	firmname_jpk	101	50
11	本社事務所所在地・住所	本社事務所所在地の住所（漢字	address	151	80
12	本社事務所郵便番号	本社事務所所在地の郵便番号	zip_code	231	7
13	本社事務所電話番号	本社事務所所在地に対応する電話番号	phone_number	238	15
14	予備		etc4	253	27
15	日経業種コード	日経が定める業種コード	nikkei_ind_code	280	6
16	法人番号	未収録時スペース	corp_number	286	13
17	予備		etc5	299	2

次に，データファイル HFSIDA920.1960.zip〜HFSIDA920.2010.2.zip を展開してできるファイルは，一行3040バイトからなる固定長フォーマットである．そのうち，先頭から100バイトは「ヘッダーパート」と呼ばれ，残りの2940バイトは「データパート」と呼ばれる（図15参照）．

ヘッダーパート ＋ データパート ＝ 一行

100バイト ＋ 2940バイト ＝ 3040バイト

なお，ヘッダーパートの「項目（名）」は，各行全てについて「共通」であることに注意しよう．

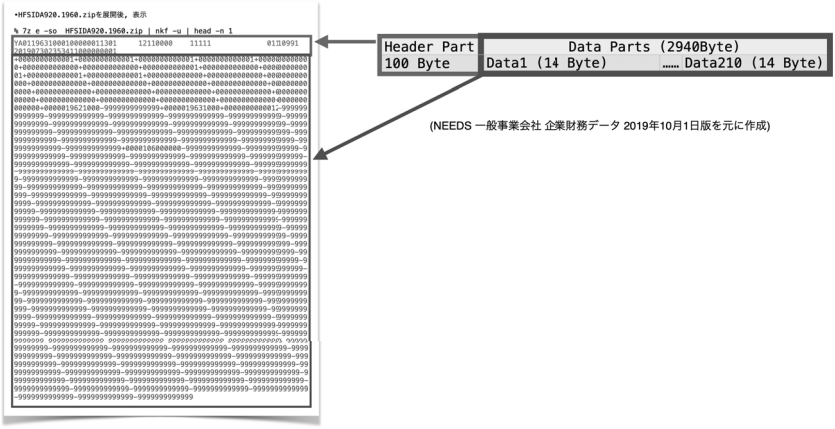
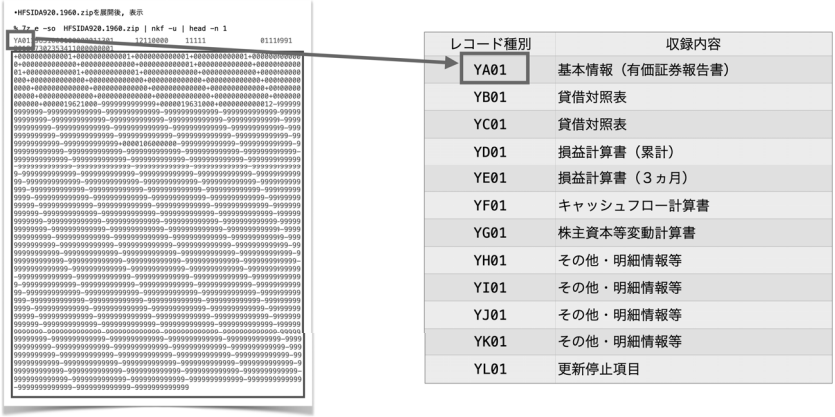


図15：データファイル **HFSIDA920.1960.zip** ～ **HFSIDA920.2010.2.zip** の仕様

ヘッダーパートに含まれる先頭4文字（例えば，YA01）は「レコード種別」とよばれ，YA01～YL01の12種類のものが用意されている．それぞれの意味については，図16を参照されたい．



レコード種別	収録内容
YA01	基本情報（有価証券報告書）
YB01	貸借対照表
YC01	貸借対照表
YD01	損益計算書（累計）
YE01	損益計算書（3ヵ月）
YF01	キャッシュフロー計算書
YG01	株主資本等変動計算書
YH01	その他・明細情報等
YI01	その他・明細情報等
YJ01	その他・明細情報等
YK01	その他・明細情報等
YL01	更新停止項目

図16：データファイル **HFSIDA920.1960.zip** ～ **HFSIDA920.2010.2.zip** のレコード種別

ヘッダーパートの仕様の詳細を表 4 に与える．この表における「形式」は「型」と「長さ」の情報を与える．例えば，"C4" の場合は，長さ 4 の文字列（character の頭文字 C）を表す．収録会社情報のファイル HFSCNM010.zip と同様に，データファイルをデータベースのテーブルとして利用するために，この情報と「カラム名」を利用する．

表 4：ヘッダーパートの仕様

項番	カラム名	形式	項目名	内容	Key 順位
1	a01	C4	レコード種別	YA0:　（有価証券報告書）基本情報 YB0:　（有価証券報告書）貸借対照表 YC0:　（有価証券報告書）貸借対照表 YD0:　（有価証券報告書）損益計算書（累計） YE0:　（有価証券報告書）損益計算書（3 ヵ月） YF0:　（有価証券報告書）キャッシュフロー計算書 YG0:　（有価証券報告書）株主資本等変動計算書 YH0:　（有価証券報告書）その他・明細情報 YI0:　（有価証券報告書）その他・明細情報 YJ0:　（有価証券報告書）その他・明細情報 YK0:　（有価証券報告書）その他・明細情報 YL0:　（有価証券報告書）更新停止項目	第 5Key
2	a02	C6	決算年月	YYYYMM 形式	第 3Key
3	a03	C2	予備		
4	a04	C1	識別フラグ	1:新規・修正 9:削除	
5	a05	C7	日経会社コード	日経が定める会社コード	第 1Key
6	a06	C4	株式コード	証券コード協議会が定める 4 桁の会社コード	
7	a07	C5	予備		
8	a08	C2	決算月数	決算月数（累計会計期間の月数を収録）	
9	a09	C1	連結・単独フラグ	1:単独 2:連結	
10	a10	C2	決算種別フラグ	10:本決算 21:第 1 四半期 24:第 4 四半期 22:第 2 四半期（中間決算） 25:第 5 四半期 23:第 3 四半期	第 4Key
11	a11	C1	決算短信情報収録フラグ	基本情報（1）収録フラグ 1:収録 0:未収録	
12	a12	C1	決算短信情報収録フラグ	基本情報（2）収録フラグ 1:収録 0:未収録	
13	a13	C1	決算短信情報収録フラグ	明細情報（1），（2）収録フラグ 1:収録 0:未収録	
14	a14	C1	予備		
15	a15	C1	予備		
16	a16	C1	予備		
17	a17	C1	予備		
18	a18	C1	有価証券報告書情報収録フラグ	基本情報（1）収録フラグ 1:収録 0:未収録	
19	a19	C1	有価証券報告書情報収録フラグ	基本情報（2）収録フラグ 1:収録 0:未収録	
20	a20	C1	有価証券報告書情報収録フラグ	明細情報（1），（2），（3）収録フラグ 1:収録 0:未収録	
21	a21	C1	有価証券報告書情報収録終了フラグ（明細情報）	1:有価証券報告書からのデータ収録のうち，明細情報（1），（2）の収録まで終了 0:未終了	
22	a22	C1	有価証券報告書情報収録終了フラグ（基本情報）	1:有価証券報告書からのデータ収録のうち，基本情報（1），（2）の収録が終了 0:未終了	
23	a23	C1	予備		
24	a24	C1	予備		
25	a25	C1	予備		
26	a26	C10	予備		
27	a27	C1	連結基準フラグ	1:日本基準 2:米国会計基準 3:国際会計基準 0:単独	第 2Key

28	a28	C1	上場フラグ	1:上場中 0:未上場・上場廃止	
29	a29	C2	上場場部	11:東証 1 部 12:東証 2 部（TOKYO PRO Market 内国を含む） 13:東証マザーズ 21:大証 1 部 22:大証 2 部 31:名証 1 部 32:名証 2 部（セントレックスを含む） 41:京都 51:広島 61:福岡（Q ボードを含む） 71:新潟 81:札幌（アンビシャスを含む） 91:ヘラクレス・スタンダード 94:ヘラクレス・グロース 99:未上場 ※上場廃止会社は，廃止時の場部	
30	a30	C1	ジャスダックフラグ	1：ジャスダック上場 0：ジャスダック未上場	
31	a31	C2	ジャスダック市場	11：ジャスダック上場 99：ジャスダック未上場 ※上場廃止会社は，廃止時の市場 ※ジャスダック取引所化以前は下記の通りに収録。 1バイト目 1:東京 2:大阪 3:名古屋の市場（1989 年 8 月以降は"1"のみ） 2 バイト目 1:上場（第一号基準銘柄，スタンダード，グロース）2:管理	
32	a32	C1	有報フラグ	1：有報提出会社 0：非有報提出会社	
33	a33	C6	予備		
34	a34	C8	データ作成日	YYYYMMDD 形式 ピーク運用時は 24 時を超え翌日が設定される場合あり	
35	a35	C6	日経業種コード	ABBCCC 形式 A:製造業;1，非製造業;2 B:日経業種中分類コード C:日経業種小分類コード	
36	a36	C1	上場情報：東京	1：1 部上場 2：2 部上場（TOKYO PRO Market 内国を含む）3：マザーズ 0：未上場	
37	a37	C1	〃：大阪	1：1 部上場 2：2 部上場 0：未上場	
38	a38	C1	〃：名古屋	1：1 部上場 2：2 部上場（セントレックスを含む）0：未上場	
39	a39	C1	〃：京都	1：1 部上場 0：未上場	
40	a40	C1	〃：広島	1：1 部上場 0：未上場	
41	a41	C1	〃：福岡	1：1 部上場（Q ボードを含む）0：未上場	
42	a42	C1	〃：新潟	1：1 部上場 0：未上場	
43	a43	C1	〃：札幌	1：1 部上場（アンビシャスを含む）0：未上場	
44	a44	C1	〃：ヘラクレス	1：スタンダード 4：グロース 0：未上場	
45	a45	C1	レコード変更フラグ	1：変更あり 0：変更なし（今回変更データの有無をレコード種別毎に収録）	
46	a46	C3	予備		

（NEEDS 企業財務データ一般事業会社，2019年10月 1 日版（日本経済新聞社，2019）を元に筆者作成）

次に，データパートのサイズは，

「データパート」サイズ＝210（項目）×14（バイト）

＝2940バイト

であり，これはレコード種別 YA01～YL01 によって変化しない．一方，項目は，レコード種別によって変化する．レコード種別 YA01～YL01 毎の収録項目の簡略化した説明を表 5 ～16に与える．なお，サイズは項目毎に14バイトに固定されており，「カラム名」はデータベースのテーブルとして利用するときに利用する．

表 5 ：YA01 基本情報（有価証券報告書）

項番	カラム名	項目名
1	b001	レコード TA0*/YA0* 収録フラグ
2	b002	レコード TB0*/YB0* 収録フラグ
3	b003	レコード TC0*/YC0* 収録フラグ
4	b004	レコード TD0*/YD0* 収録フラグ
5	b005	レコード TE0*/YE0* 収録フラグ
6	b006	レコード TF0*/YF0* 収録フラグ
7	b007	レコード TG0*/YG0* 収録フラグ
8	b008	レコード TH0*/YH0* 収録フラグ
9	b009	レコード TI0*/YI0* 収録フラグ
10	b010	レコード TJ0*/YJ0* 収録フラグ
11	b011	レコード TK0*/YK0* 収録フラグ
12	b012	レコード TL0*/YL0* 収録フラグ
⋮	⋮	⋮
209	b209	予備
210	b210	予備

表 6 ：YB01（貸借対照表）

項番	カラム名	項目名
1	b001	資産合計
2	b002	予備
3	b003	予備
4	b004	予備
5	b005	予備
6	b006	予備
7	b007	予備
8	b008	予備
9	b009	予備
10	b010	予備
11	b011	予備
12	b012	予備
⋮	⋮	⋮
209	b209	予備
210	b210	予備

表 7 ：YC01（貸借対照表）

項番	カラム名	項目名
1	b001	負債
2	b002	純資産
3	b003	資本金
4	b004	自己資本
5	b005	予備
6	b006	予備
7	b007	予備
8	b008	予備
9	b009	予備
10	b010	予備
11	b011	予備
12	b012	予備
⋮	⋮	⋮
209	b209	予備
210	b210	予備

表 8 ：YD01（損益計算書（累計））

項番	カラム名	項目名
1	b001	売上高・営業収益
2	b002	営業利益
3	b003	経常利益
4	b004	親会社株主に帰属する当期純利益（連結）／当期利益（単独）
5	b005	売上高・営業収益（短信サマリー）
6	b006	予備
7	b007	予備
8	b008	予備
9	b009	予備
10	b010	予備
11	b011	予備
12	b012	予備
⋮	⋮	⋮
209	b209	予備
210	b210	予備

表 9 ：YE01（損益計算書（ 3 ヵ月））

項番	カラム名	項目名
1	b001	売上高・営業収益
2	b002	営業利益
3	b003	経常利益
4	b004	親会社株主に帰属する当期純利益（連結）／当期利益（単独）
5	b005	予備
6	b006	予備
7	b007	予備
8	b008	予備
9	b009	予備
10	b010	予備
11	b011	予備
12	b012	予備
⋮	⋮	⋮
209	b209	予備
210	b210	予備

表10：YF01（キャッシュフロー計算書）

項番	カラム名	項目名
1	b001	営業活動によるキャッシュフロー
2	b002	投資活動によるキャッシュフロー
3	b003	財務活動によるキャッシュ・フロー
4	b004	現金および現金同等物の期末残高
5	b005	予備
6	b006	予備
7	b007	予備
8	b008	予備
9	b009	予備
10	b010	予備
11	b011	予備
12	b012	予備
⋮	⋮	⋮
209	b209	予備
210	b210	予備

表11：YG01（株主資本等変動計算書）

項番	カラム名	項目名
1	b001	【資本金】当期首残高
2	b002	新株の発行
3	b003	資本金から資本準備金またはその他資本剰余金への振替
4	b004	資本準備金から資本金への振替
5	b005	その他資本剰余金から資本金への振替
6	b006	企業結合または会社分割による増減
7	b007	連結範囲または持分法適用範囲の変動による増減
8	b008	その他の資本金増減
9	b009	当期変動額合計
10	b010	当期末残高
11	b011	【資本準備金】当期首残高
12	b012	新株の発行
⋮	⋮	⋮
209	b209	予備
210	b210	予備

表12：YH01（その他・明細情報等）

項番	カラム名	項目名
1	b001	受取手形割引高
2	b002	受取手形裏書譲渡高
3	b003	貸倒引当金（欄外注記分）
4	b004	税効果会計の対象となった繰越欠損金
5	b005	減価償却実施額（有形無形その他の合計）≪累計≫
6	b006	減価償却実施額（有形無形その他の合計）≪3カ月≫
7	b007	うち有形固定資産減価償却実施額≪累計≫
8	b008	うち有形固定資産減価償却実施額≪3カ月≫
9	b009	減損損失（有形無形その他の合計）
10	b010	うち有形固定資産減損損失
11	b011	減価償却累計額・減損損失累計額控除前（有形無形その他合計）
12	b012	うち有形固定資産減価償却累計額・減損損失累計額控除前
⋮	⋮	⋮
209	b209	【年金資産の内訳】一般勘定（割合）
210	b210	【年金資産の内訳】その他（割合）

表13：YI01（その他・明細情報等）

項番	カラム名	項目名
1	b001	売買目的有価証券貸借対照表計上額リース
2	b002	売買目的有価証券損益に含まれた評価差額リース
3	b003	満期保有目的債券合計貸借対照表計上額リース
4	b004	満期保有目的債券合計時価リース
5	b005	満期保有目的債券合計差額リース
6	b006	うち時価が計上額を超える分リース
7	b007	うち時価が計上額を超えない分リース
8	b008	子会社株式貸借対照表計上額リース
9	b009	子会社株式時価リース
10	b010	子会社株式差額リース
11	b011	関連会社株式貸借対照表計上額リース
12	b012	関連会社株式時価リース
⋮	⋮	⋮
209	b209	【退職給付債務調整表】期末における退職給付債務リース
210	b210	退職給付信託額の割合リース

表14：YJ01（その他・明細情報等）

項番	カラム名	項目名
1	b001	１単元の株数
2	b002	上位十大株主持株数
3	b003	少数特定者持株数
4	b004	浮動株数
5	b005	大株主・役員以外の株主数
6	b006	投信持株数
7	b007	年金持株数
8	b008	役員持株数
9	b009	従業員持株会持株数
10	b010	授権株式数
11	b011	政府公共団体所有株数
12	b012	金融機関所有株数
⋮	⋮	⋮
209	b209	予備
210	b210	予備

表15：YK01（その他・明細情報等）

項番	カラム名	項目名
1	b001	商品・製品売上高
2	b002	製品売上高
3	b003	商品売上高
4	b004	その他売上高・営業収益・営業収入
5	b005	（▲）売上値引・戻り高
6	b006	金融収益
7	b007	売上高・営業収益合計
8	b008	期首製品・商品棚卸高
9	b009	当期製品製造原価
10	b010	当期商品仕入高
11	b011	小計
12	b012	期末製品・商品棚卸高
⋮	⋮	⋮
209	b209	予備
210	b210	予備

表16：YL01（更新停止項目）

項番	カラム名	項目名
1	b001	前期繰越利益
2	b002	うち合併引継未処分利益
3	b003	利益準備金取崩額
4	b004	諸任意積立金目的取崩額
5	b005	自己株式処分差損
6	b006	自己株式消却額
7	b007	普通株式中間配当額
8	b008	優先株式中間配当額
9	b009	中間配当に伴う利益準備金積立額
10	b010	その他諸任意積立金目的取崩額
11	b011	中間配当積立金取崩額
12	b012	当期未処分利益
⋮	⋮	⋮
209	b209	予備
210	b210	予備

B. 2 データベースの構築

IV 節でも述べたが，この製品（NEEDS 企業財務データ（一般事業会社）2019年版）に関しては[23]，2010年代初めに以下のような仕様変更があった．

NEEDS 企業財務データの仕様変更

- 変更前は，単独決算と連結決算のファイル群を分けて販売されており，会計基準に関する分類もなされていなかった．
- 仕様変更に伴って，単独決算と連結決算のデータが合併され，会計基準として「日本基準」，「米国会計基準」，「国際会計基準」のデータも同一のファイルに合併されて配布されるようになった．
- 本決算と中間決算以外に四半期決算の情報も収録されるようになった．
- データを一意化するためのキー（主キー）は，従来のものが（日経会社コード，決算年月日）の 2 個であったのに対して，表17の 5 個に増加した．

表17：NEEDS 企業財務データに関する主キー

キー順位	カラム名	形式	項目	収録内容
第 1 キー	a05	C7	日経会社コード	日経が定める会社コード
第 2 キー	a27	C1	連結基準フラグ	1：日本基準，2：米国会計基準， 3：国際会計基準，0：単独
第 3 キー	a02	C6	決算年月	YYYYMM 形式
第 4 キー	a10	C2	決算種別フラグ	10：本決算，21：第 1 四半期， 22：第 2 四半期（中間決算），23：第 3 四半期， 24：第 4 四半期，25：第 5 四半期
第 5 キー	a01	C4	レコード種別	決算短信情報レコード：TA01〜TL01， 有価証券報告書情報レコード：YA01〜YL01

これらの仕様変更から，データベースとデータ抽出環境を構築することに関して以下のような問題が生じた．

23）2010年までは，NEEDS 企業財務データ（一般事業会社），MT 版という名称で販売されていた．ここで，MT とは配布媒体が Magnetic Tape であることを指していたが，2008年には DVD などのメディアに変更されていた．

仕様変更に伴うデータベース及びデータ抽出環境構築に関する問題

（CP1）SQL 問合せによるデータ抽出時間の問題
（CP2）一意性の確保の問題

問題（CP1）については，以前と同様の方法で実験的にデータベース環境を構築し，これまで利用していた適当な SQL 問合せ（ある時点での売上高や資産合計など 2，3 の指標に対するデータを抽出するためのスクリプト）による抽出を試みたところ，データの規模が大きくなったことから，数十分（場合によっては数時間）を要する場合があった．さらに，問題（CP2）に起因する問題から，主キーをいくつも与えないと企業の複数の決算にもとづく結果が抽出されてしまい，一意に抽出できない．これらの問題に対して，以下のような解決策を講じた．

解決策

（S1）主キーによるデータベースの分離
（S2）SQL 問合せの工夫

解決策（S1）について説明する．なお，主キーの説明については，表17を参照されたい．まず，今回利用してるデータファイルは，第 5 キーの「レコード種別」における「有価証券報告書情報レコード」のみであり，「決算短信情報レコード」のものは利用しない．このことから，第 5 キーは検討の対象外となる．次に，これまでに，構築してきた財務データベースは，「本決算」かつ，「連結」または「単独」のものであるので，第 4 キーである「決算種別フラグ」として「本決算」（10）を選択し，かつ，第 2 キーである「連結基準フラグ」を「単独」（0）とそれ以外（「日本基準」1，「米国会計基準」2，「国際会計基準」3）に分けて選択することによって，主キーを第 1 キーである「日経会社コード」と第 3 キーである「決算年月」に絞り込むことができる．つまり，「連結本決算」と「単独本決算」のレコードを抽出し，別のデータベースへ分離することによって，サイズを縮小することが可能となる．

なお，この方法は，ユーザに多くの主キーを指定させることなく一意性が確保できるという「副次的効果」（side effect）も与える．以上のことから，解決策（S1）はサーバサイドの対応といえる．

次に，解決策（S2）について説明する．これまで運用してきた財務データベースは，各テーブルのサイズが（今回のものよりも）小さかったことから，JOIN を実行してテーブルをフルに結合しても，それにかかる時間は，それほど気になるものではなかったが，今回実験的に構築したデータベース環境で試行してみたところ，「現実的な時間」で結合することが難しいことがわかった．そこで，結合する際に，テーブルから事前にカラムを指定したものをテーブル化して，結合時のテーブルサイズを縮小するというアイデアを利用したところ，現実的な時間で抽出することが可能であることがわかった．このアイデアは，データベースの分野ではよく知られた単純な方法であり，クライアント（ユーザ）サイドの解決策といえる．

サーバサイドの解決策（P1）を考慮し，以下のような手順でデータベースの構築を行った：

NEEDS 企業財務データ（一般事業会社）2019年版にもとづくデータベースの構築手順

（NS1）データファイル HFSIDA920.1960.zip〜HFSIDA920.2010.2.zip と収録会社情報のファイル HFSCNM010.zip の前処理
（NS2）データベース needs2019 を作成し，テーブル ya01〜yl01（レコード種別 YA01〜YL01 に対応）を作成後，前処理されたデータファイル YA01tab.data〜YL01tab.data からテーブル ya01〜yl01 へロード
（NS3）データベース needs2019 にテーブル firmlist を作成し，ファイル firmlist.txt からロード
（NS4）データベース needs2019 から，連結本決算のデータベース needs2019cons と単独本決算のデータベース needs2019uncons を分離・作成

上記の手順は，OS として，macOS と Ubuntu(Linux)，さらに，RDBMS として，MySQL と PostgreSQL の合計 4 種類の環境（MAMP，MAPP，LAMP，LAPP）について実施する必要がある（表 1 と図 1 参照）．

以下に，それぞれの環境のもとでの手順を確認していく．なお，macOS

環境で構築に利用したデータファイル，スクリプトファイルのディレクトリ構成については図17を参照されたい[24)]

```
MT-general2019-macOS
├── DBMaking
│   ├── Makefile
│   ├── dumpfromPostgreSQLgen-cons.sql
│   ├── dumpfromPostgreSQLgen-uncons.sql
│   ├── loadfirmlisttoMySQLgen.sql
│   ├── loadfirmlisttoPostgreSQLgen-cons.sql
│   ├── loadfirmlisttoPostgreSQLgen-uncons.sql
│   ├── loadfirmlisttoPostgreSQLgen.sql
│   ├── loadtoMySQLgen.sql
│   ├── loadtoPostgreSQLgen-cons.sql
│   ├── loadtoPostgreSQLgen-uncons.sql
│   ├── loadtoPostgreSQLgen.sql
│   ├── mkcons.sql
│   ├── mkuncons.sql
│   ├── replacedumpPostgreSQL-CDIR-cons.sh
│   ├── replacedumpPostgreSQL-CDIR-uncons.sh
│   ├── replaceloadCDIR-cons.sh
│   ├── replaceloadCDIR-uncons.sh
│   ├── replaceloadCDIR.sh
│   ├── replaceloadPostgreSQL-CDIR-cons.sh
│   ├── replaceloadPostgreSQL-CDIR-uncons.sh
│   ├── replaceloadPostgreSQL-CDIR.sh
│   ├── replaceloadfirmlist.sh
│   ├── replaceloadfirmlistPostgreSQL-cons.sh
│   ├── replaceloadfirmlistPostgreSQL-uncons.sh
│   ├── replaceloadfirmlistPostgreSQL.sh
│   ├── script-p.sh
│   ├── scriptfirmlist.sh
│   ├── separate.pl
│   ├── separatefirmlist.pl
│   ├── seqscript-p.sh
│   └── unzip.sh
└── datafiles
    ├── HFSCNM010.zip
    ├── HFSIDA920.1960.zip
    ├── HFSIDA920.1970.zip
    ├── HFSIDA920.1980.zip
    ├── HFSIDA920.1990.zip
    ├── HFSIDA920.2000.1.zip
    ├── HFSIDA920.2000.2.zip
    ├── HFSIDA920.2010.1.zip
    └── HFSIDA920.2010.2.zip
```

図17：macOS 環境用のデータファイル，スクリプトファイルのディレクトリ構成

24）Ubuntu 環境用のディレクトリ・ファイルの構成はほぼ同じである．

B. 2. 1 前処理

ファイルの前処理（NS1）は，4 種類の環境（MAMP，MAPP，LAMP，LAPP）で共通のスクリプトで処理が可能である．具体的には以下の手順で前処理が行われる：

前処理の手順

（PP1）データの ZIP ファイル HFSIDA920.1960.zip〜HFSIDA920.2010.2.zip を展開し，一つのファイル full.txt へ結合

（PP2）ファイル full.txt をレコード種別ごとのファイル YA01.data〜YL01.data へ分離し，固定長フォーマットをデータの仕様に応じて，適切にレコード間に分割記号（タブ区切り）を挿入後，文字コードを CP932 から UTF-8 へ変換することによって，ファイル YA01.data〜YL01.data をファイル YA01tab.data〜YL01tab.data へ変換

（PP3）会社リストの ZIP ファイル HFSCNM010.zip をファイル fullfirmlist.txt へ展開し，定長フォーマットをデータの仕様に応じて，適切にレコード間に分割記号（タブ区切り）を挿入後，文字コードを CP932 から UTF-8 へ変換することによって，ファイル fullfirmlist.txt をファイル firmlist.txt へ変換

これらの手順を実行するスクリプトを Makefile のターゲット preprocess に記述した（スクリプト 5 参照）．スクリプト 5 において，2，6 行目は処理時間を計測するための指定である．

スクリプト 5 ：Makfile: ターゲット preprocess

```
1 preprocess:
2       date > start-preprocess.txt
3       /bin/bash unzip.sh
4       /bin/bash seqscript-p.sh
5       /bin/bash scriptfirmlist.sh
6       date > end-preprocess.txt
```

スクリプト 5 の 3，4，5 行目の各シェルスクリプトが前処理の手順（PP1），（PP2），（PP3）を実現するためのものである．

まず，手順（PP1）を実現するための unzip.sh はスクリプト 6 で与えられる．

スクリプト 6 ：unzip.sh

```
#!/bin/bash
echo Start unzip!
7z e -so ../datafiles/HFSIDA920.1960.zip >  full.txt
7z e -so ../datafiles/HFSIDA920.1970.zip >> full.txt
7z e -so ../datafiles/HFSIDA920.1980.zip >> full.txt
7z e -so ../datafiles/HFSIDA920.1990.zip >> full.txt
7z e -so ../datafiles/HFSIDA920.2000.1.zip >> full.txt
7z e -so ../datafiles/HFSIDA920.2000.2.zip >> full.txt
7z e -so ../datafiles/HFSIDA920.2010.1.zip >> full.txt
7z e -so ../datafiles/HFSIDA920.2010.2.zip >> full.txt
echo Complete unzip!
```

このシェルスクリプトでは，7z コマンド[25] を用いて ZIP ファイルを展開した後，リダイレクション（>, >>）機能を使って，一つのテキストファイル full.txt に結合している．

次に，手順（PP2）を実現するための seqscript-p.sh は，スクリプト 7 で与えられる．

スクリプト 7 ：seqscript-p.sh

```
#!/bin/bash
for i in {A..L}
do
echo "Y"$i"01"
/bin/bash script-p.sh "Y"$i"01"
done
```

このシェルスクリプトでは，2 行目で変数 i を A から L まで変化させ，3 行目から 6 行目で，順次文字列 "Y"$i"01" を生成し，シェルスクリプト script-p.sh を繰り返し適用している．例えば，i に A が代入されている場合は，（文字列）YA01 が生成され，スクリプト script-p.sh の引数として与えられる． 5 行目で使用されている処理手順を実行するスクリプト script-p.sh は，スクリプト 8 で与えられる．

スクリプト 8 ：script-p.sh

```
#!/bin/bash
export LC_ALL=C
#
parallel --pipepart -k --block 100M -a "full.txt" 'grep' $1 > $1".data"
```

25）https://www.7-zip.org/

```
#
echo Complete Separate!
#
perl separate.pl $1".data"
parallel --pipepart -k --block 100M -a "temp" "sed␣'/^$/d'" > $1".txt"
parallel --pipepart -k --block 100M -a $1".txt" "nkf␣--utf8" > $1"tab.data"
rm temp
#
echo Complete Tables!
echo Finish!
```

このシェルスクリプトでは，4行目で全データを結合したテキストファイル full.txt から，「親」スクリプトから受け取った引数 $1（例えば，YA01）を含む行を grep コマンドを使って抽出し，ファイル（例えば，YA01.data）に書き出している．その際，この処理を GNU parallel[26] で並列化することによって処理時間を短縮している．次に，8行目で perl[27] を利用して，分離された個々のファイルの適切な箇所に分割記号を挿入している．その際，引数として分離された個々のファイル名を与え，Perl スクリプトファイル separate.pl（スクリプト9参照）を実行している．

スクリプト9：separate.pl（一部抜粋）

```
#!/usr/bin/perl
#####
open(IN,"@ARGV");
open(OUT,">temp");
#####
do{
read(IN,$xx,3042);
$a[1]=substr($xx,0,4);
$a[2]=substr($xx,4,6);
: (中略)
$a[46]=substr($xx,97,3);
$b[1]=substr($xx,100,14);
$b[2]=substr($xx,114,14);
: (中略)
$b[210]=substr($xx,3026,14);
print OUT
"$a[1]\t$a[2]\t...\t$a[46]\t$b[1]\t$b[2]\t...\t$b[210]\n";
} while (eof(IN)!=1);
close (IN);
```

このスクリプトの，3行目では，引数（"@ARGV"）として与えられたファ

26）https://www.gnu.org/software/parallel/
27）https://www.perl.org/

イルを入力（IN）とし，1行づつ読み取りながら，適切に文字列を切り出したもの（8行目から15行目）にタブコード（\t）を挿入し，出力するということを最終行まで繰り返し，最終的にファイル temp に出力している（4行目）．この工程で出力されたファイル temp を，スクリプト8の9行目で空行を取り除いた後，10行目で nkf[28] コマンド用いて文字コードを UTF-8 に変換したものを最終的にファイル YA01tab.data〜YL01tab.data として出力している．なお，これらの工程は GNU parallel を用いて並列化することによって処理時間を短縮している．

前処理の最後の手順（PP3）を実現するための scriptfirmlist.sh は，スクリプト10で与えられる．

スクリプト10：scriptfirmlist.sh

```
 1 export LC_ALL=C
 2 #
 3 echo Start unzip!
 4 7z e -so ../datafiles/HFSCNM010.zip > fullfirmlist.txt
 5 #
 6 echo Complete unzip!
 7 #
 8 perl separatefirmlist.pl fullfirmlist.txt
 9 sed '/^$/d' temp | nkf --utf8 > firmlist.txt
10 rm temp;
11 #
12 echo Complete Tables!
13 #
14 rm  fullfirmlist.txt
15 #
16 echo Finish!
```

このシェルスクリプトの4行目で，7z コマンドを用いて収録会社に関する情報が納められた ZIP ファイル HFSCNM010.zip を展開した後，リダイレクション（>）機能を使って，テキストファイル fullfirmlist.txt に出力している．8行目で perl を利用して，分離された個々のファイルの適切な箇所に分割記号を挿入している．その際，引数としてファイル名 fullfirmlist.txt を与え，Perl スクリプトファイル separatefirmlist.pl

28）https://osdn.net/projects/nkf/

（スクリプト11参照）を実行している．

スクリプト11：separatefirmlist.pl

```
#!/usr/bin/perl
#####
open(IN,"@ARGV");
open(OUT,">temp");
#####
do{
read(IN,$xx,302);
$a[1]=substr($xx,0,4);
$a[2]=substr($xx,4,7);
$a[3]=substr($xx,11,4);
$a[4]=substr($xx,15,1);
$a[5]=substr($xx,16,9);
$a[6]=substr($xx,25,4);
$a[7]=substr($xx,29,11);
$a[8]=substr($xx,40,30);
$a[9]=substr($xx,70,30);
$a[10]=substr($xx,100,50);
$a[11]=substr($xx,150,80);
$a[12]=substr($xx,230,7);
$a[13]=substr($xx,237,15);
$a[14]=substr($xx,252,27);
$a[15]=substr($xx,279,6);
$a[16]=substr($xx,285,13);
$a[17]=substr($xx,298,2);
#
print OUT
"$a[1]\t$a[2]\t$a[3]\t$a[4]\t$a[5]\t$a[6]\t$a[7]\t$a[8]\t$a[9]\t$a[10]\t$a
    [11]\t$a[12]\t$a[13]\t$a[14]\t$a[15]\t$a[16]\t$a[17]\n";
} while (eof(IN)!=1);
close (IN);
```

このスクリプトの，3行目では，引数（"@ARGV"）として与えられたファイル fullfirmlist.txt を入力（IN）とし，1行づつ読み取りながら，適切に文字列を切り出したもの（8行目から24行目）にタブコード（\t）を挿入し，出力するということを最終行まで繰り返し（26行目から28行目），最終的にファイル temp に出力している（4行目）．この工程で出力されたファイル temp を，スクリプト10の9行目で sed コマンドを用いて空行を取り除いた後，さらに nkf コマンド用いて文字コードを UTF-8 に変換したものを最終的にファイル firmlist.txt として出力している．

以上の前処理の流れを簡略化して可視化したものを図18に与える．

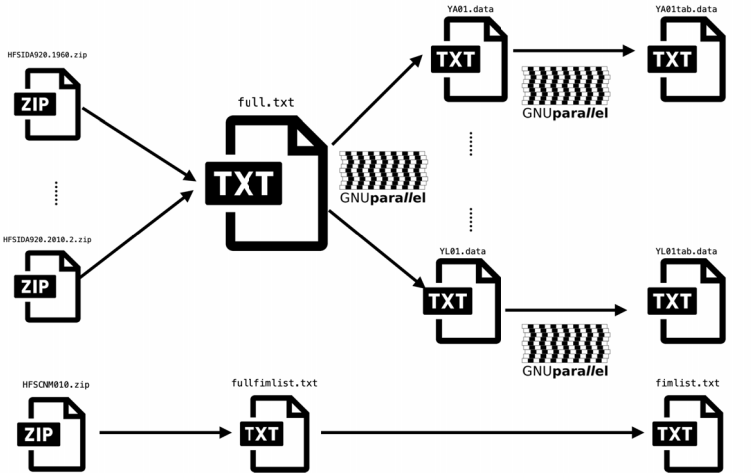


図18：前処理

なお，前処理は，ディレクトリ DBMaking（図17参照）をカレント（ディレクトリ）とし，ターミナル（コマンドライン）上で以下のように入力することによって実行できる（ただし，%，$ はシェルプロンプトである）．

ターゲット preprocess の実行: macOS，Ubuntu 共通

```
% make preprocess
```

この処理時間は，スクリプト 5 において，2，6 行目の実行結果を比較することによってわかる．macOS 上で実行した結果を以下に与える．

macOS 上でのターゲット preprocess の処理時間の計測

```
% cat start-preprocess.txt
Sun Sep  6 08:46:40 JST 2020
% cat end-preprocess.txt
Sun Sep  6 08:58:06 JST 2020
```

この結果から，11分26秒であることがわかる[29]．

一方，Ubuntu 上で実行した結果も以下に与える．

Ubuntu 上でのターゲット preprocess の処理時間の計測

```
$ cat start-preprocess.txt
Wed Aug 19 16:43:32 JST 2020
$ cat end-preprocess.txt
Wed Aug 19 16:52:15 JST 2020
```

この結果から，8分43秒である[30]．

B. 2. 2 データベース needs2019，needs2019cons，needs2019uncons の構築

ここでは，構築手順（NS2），（NS3），（NS4）におけるデータベース needs2019（全体），needs2019cons（連結本決算），needs2019uncons（単独本決算）の構築工程を詳細に述べる．その際，RDBMS として，MySQL と PostgreSQL を利用する場合に分けて考え，さらに macOS と Ubuntu の間で差異がある場合については適宜場合わけする．

なお，最終的に構築したデータベースに対しては，適当なユーザを作成し，アクセスする権限を与える必要があるが，セキュリティの観点から，それらの設定に関する詳細は割愛する．

■ MySQL の場合

RDBMS として，MySQL を利用する場合は，macOS と Ubuntu（すなわち，MAMP と LAMP）の間で構築するためのスクリプトに関して差異はない[31]．

構築手順（NS2），（NS3），（NS4）を実行するスクリプトを Makefile のターゲット MSDB に記述した（スクリプト12参照）．スクリプト12において，2行目と9行目は処理時間を計測するための指定である．

29）この結果は，MacBook Pro 2018 で計測したものである．
30）この結果は，MacBook Pro 2018 上で VMware によって構築した Ubuntu 18.04 の仮想環境で計測したものである．
31）データベースを構築するためのスクリプトの差異をなくすためには，MySQL のバージョンに依存して，macOS と Ubuntu 上での環境設定に関して調整が必要となる．ここでは，macOS にはバージョン5.6，Ubuntu には5.7をインストールして運用している．

スクリプト12：Makefile：ターゲット MSDB

```
1 MSDB:
2       date > start-MSDB.txt
3       /bin/bash replaceloadCDIR.sh
4       mysql -u root  --local_infile=1 -p******** < ./loadtoMySQL.sql
5       /bin/bash replaceloadfirmlist.sh
6       mysql -u root  --local_infile=1 -p******** < ./loadfirmlisttoMySQL.
           sql
7       mysql -u root --local_infile=1 -p******** < ./mkcons.sql
8       mysql -u root --local_infile=1 -p******** < ./mkuncons.sql
9       date > end-MSDB.txt
```

スクリプト12の 3，4 行目が構築手順（NS2）を実現するためのものであり，5，6 行目によって構築手順（NS3）が，さらに 7，8 行目の指定で構築手順（NS4）が実現する．

まず，構築手順（NS2）を実現するためのスクリプトについてみる． 3 行目で利用されているシェルスクリプト replaceloadCDIR.sh の内容（スクリプト13）を見ると， 2 行目でカレントディレクトリの情報を環境変数に代入（CDIR=$PWD）しており， 3 行目では sed コマンドを利用して，SQL スクリプトファイル loadtoMySQLgen.sql（スクリプト14）における文字列 CDIR をカレントディレクトリの情報で置換（スクリプト14の20〜22行目）し，その結果を SQL スクリプトファイル loadtoMySQL.sql にリダイレクション（>）機能を使って出力している．この意味で，SQL スクリプトファイル loadtoMySQL.sql は，シェルスクリプト replaceloadCDIR.sh によって SQL スクリプトファイル loadtoMySQLgen.sql から生成（generate）される（図19参照）．

スクリプト13：replaceloadCDIR.sh

```
1 #!/bin/bash
2 CDIR=$PWD
3 sed -e "s|CDIR|$CDIR|g" loadtoMySQLgen.sql > loadtoMySQL.sql
```

図19：SQL スクリプトファイル **loadtoMySQL.sql** の生成

スクリプト14：loadtoMySQLgen.sql（一部抜粋）

```
 1 DROP DATABASE IF EXISTS needs2019;
 2 CREATE DATABASE needs2019;
 3 USE needs2019
 4 DROP TABLE IF EXISTS ya01;
 5 : (中略)
 6 DROP TABLE IF EXISTS yl01;
 7 CREATE TABLE ya01 (
 8 a01 VARCHAR(4),
 9 a02 INT(6),
10 : (中略)
11 a46 VARCHAR(3),
12 b001 VARCHAR(14),
13 b002 VARCHAR(14),
14 : (中略)
15 b210 VARCHAR(14)
16 );
17 CREATE TABLE yb01 LIKE ya01;
18 : (中略)
19 CREATE TABLE yl01 LIKE ya01;
20 LOAD DATA LOCAL INFILE 'CDIR/YA01tab.data' INTO TABLE ya01 FIELDS
       TERMINATED BY '\t';
21 : (中略)
22 LOAD DATA LOCAL INFILE 'CDIR/YL01tab.data' INTO TABLE yl01 FIELDS
       TERMINATED BY '\t';
```

生成された SQL スクリプトファイル loadtoMySQL.sql は，データベース needs2019 と，テーブル ya01〜yl01 を作成（2〜19行目）した後，データファイル YA01tab.data〜YL01tab.data から，テーブル ya01〜yl01 へデータをロードするためのものであり，実際にターゲット MSDB（スクリプト12）の 4 行目で実行される．

次に，構築手順（NS3）を実現するためのスクリプトとして，Makefile のターゲット MSDB（スクリプト12）の 5 行目で実行されるスクリプトファ

イル replaceloadfirmlist.sh（スクリプト15）は，replaceloadCDIR.sh（スクリプト13）と同様の役割を果たす．つまり，2行目でカレントディレクトリの情報を環境変数に代入（CDIR=$PWD）し，3行目で sed コマンドを利用して，SQL スクリプトファイル loadfirmlisttoMySQLgen.sql（スクリプト16）における文字列 CDIR をカレントディレクトリの情報で置換（スクリプト16の22行目）し，その結果を SQL スクリプトファイル loadfirmlisttoMySQL.sql にリダイレクション（>）機能を使って出力している．この意味で，SQL スクリプトファイル loadfirmlisttoMySQL.sql は，シェルスクリプト replaceloadfirmlist.sh によって SQL スクリプトファイル loadfirmlisttoMySQLgen.sql から生成される（図20参照）．

スクリプト15：replaceloadfirmlist.sh

```
1 #!/bin/bash
2 CDIR=$PWD
3 sed -e "s|CDIR|$CDIR|g" loadfirmlisttoMySQLgen.sql > loadfirmlisttoMySQL.
    sql
```

図20：SQL スクリプトファイル loadfirmlisttoMySQL.sql の生成

スクリプト16：loadfirmlisttoMySQLgen.sql

```
1 USE needs2019;
2 DROP TABLE IF EXISTS firmlist;
3 CREATE TABLE firmlist (
4 record_type VARCHAR(4),
5 nikkei_corp_code VARCHAR(7),
6 stock_code VARCHAR(4),
7 etc1 VARCHAR(1),
8 etc2 VARCHAR(9),
9 finance_code VARCHAR(4),
```

```
10 etc3 VARCHAR(11),
11 firmname VARCHAR(30),
12 firmname_jp VARCHAR(30),
13 firmname_jpk VARCHAR(50),
14 address VARCHAR(80),
15 zip_code VARCHAR(7),
16 phone_number VARCHAR(15),
17 etc4 VARCHAR(27),
18 nikkei_ind_code VARCHAR(6),
19 corp_number VARCHAR(13),
20 etc5 VARCHAR(2)
21 );
22 LOAD DATA LOCAL INFILE 'CDIR/firmlist.txt' INTO TABLE firmlist FIELDS
       TERMINATED BY '\t';
```

生成されたSQLスクリプトファイルloadfirmlisttoMySQL.sqlは，データベースneeds2019においてテーブルfirmlistを作成（3〜21行目）した後，収録企業に関する情報が収められたファイルfirmlist.txtから，テーブルfirmlistへデータをロードするためのものであり，実際にターゲットMSDB（スクリプト12）の6行目で実行される．

最後に，構築手順（NS4）を実現するためのスクリプトをみる．Makefileのターゲット MSDB（スクリプト12）の7行目で指定されているSQLスクリプトファイルmkcons.sql（スクリプト17）は，データベースneeds2019から，連結本決算のデータベースneeds2019consを分離するためのものである．また，8行目で指定されているSQLスクリプトファイルmkuncons.sql（スクリプト18）は，データベースneeds2019から，連結本決算のデータベースneeds2019unconsを分離するためのものである．これらのスクリプトは，ほぼ機能が同じであるので，SQLスクリプトファイルmkcons.sql（スクリプト17）を主に説明する．まず，2行目で連結本決算のデータベースneeds2019consを作成した後，テーブルya01〜yl01を作成している（7行目から19行目）．さらに，収録企業のリストのテーブルfirmlistを作成（22行目から40行目）し，41行目で既存のデータベースのテーブルneeds2019.ya01の全ての列（*）から，SQL問合せに関するWHERE句に連結本決算を満たす条件を与え，制約付きでレコードを抽出し，新しいデータベースのテーブルneeds2019cons.ya01へ挿入

している．ここで，

(WHERE 句条件 1)　WHERE a09='2' AND a10='10' AND a27 !='0'

は，ヘッダーパートの仕様（表 4 ）から，「連結・単独フラグ」a09 として 2（連結）を選択し，かつ「決算別フラグ」a10 として 10（本決算），さらに，「連結基準フラグ」a27 として 0（単独）でないものを選択していることがわかる．

スクリプト17：mkcons.sql

```
DROP DATABASE IF EXISTS needs2019cons;
CREATE DATABASE needs2019cons;
USE needs2019cons
DROP TABLE IF EXISTS ya01;
: (中略)
DROP TABLE IF EXISTS yl01;
CREATE TABLE ya01 (
a01 VARCHAR(4),
a02 INT(6),
: (中略)
a46 VARCHAR(3),
b001 VARCHAR(14),
b002 VARCHAR(14),
: (中略)
b210 VARCHAR(14)
);
CREATE TABLE yb01 LIKE ya01;
: (中略)
CREATE TABLE yl01 LIKE ya01;
#
DROP TABLE IF EXISTS firmlist;
CREATE TABLE firmlist (
record_type VARCHAR(4),
nikkei_corp_code VARCHAR(7),
stock_code VARCHAR(4),
etc1 VARCHAR(1),
etc2 VARCHAR(9),
finance_code VARCHAR(4),
etc3 VARCHAR(11),
firmname VARCHAR(30),
firmname_jp VARCHAR(30),
firmname_jpk VARCHAR(50),
address VARCHAR(80),
zip_code VARCHAR(7),
phone_number VARCHAR(15),
etc4 VARCHAR(27),
nikkei_ind_code VARCHAR(6),
corp_number VARCHAR(13),
etc5 VARCHAR(2)
);
INSERT INTO needs2019cons.ya01 SELECT * FROM needs2019.ya01 WHERE a09='2'
```

```
    AND a10='10' AND a27!='0';
: (中略)
INSERT INTO needs2019cons.yl01 SELECT * FROM needs2019.yl01 WHERE a09='2'
    AND a10='10' AND a27!='0';
INSERT INTO needs2019cons.firmlist SELECT * FROM needs2019.firmlist;
```

同様の処理をテーブル yl01 まで行い（43行目），最後に firmlist については全く同じテーブルをデータベース needs2019 から needs2019cons へ挿入している．

スクリプト18：mkuncons.sql

```
DROP DATABASE IF EXISTS needs2019uncons;
CREATE DATABASE needs2019uncons;
USE needs2019uncons
DROP TABLE IF EXISTS ya01;
: (中略)
DROP TABLE IF EXISTS yl01;
CREATE TABLE ya01(
a01 VARCHAR(4),
a02 INT(6),
: (中略)
a46 VARCHAR(3),
b001 VARCHAR(14),
b002 VARCHAR(14),
: (中略)
b210 VARCHAR(14)
);
CREATE TABLE yb01 LIKE ya01;
: (中略)
CREATE TABLE yl01 LIKE ya01;
#
DROP TABLE IF EXISTS firmlist;
CREATE TABLE firmlist (
record_type VARCHAR(4),
nikkei_corp_code VARCHAR(7),
stock_code VARCHAR(4),
etc1 VARCHAR(1),
etc2 VARCHAR(9),
finance_code VARCHAR(4),
etc3 VARCHAR(11),
firmname VARCHAR(30),
firmname_jp VARCHAR(30),
firmname_jpk VARCHAR(50),
address VARCHAR(80),
zip_code VARCHAR(7),
phone_number VARCHAR(15),
etc4 VARCHAR(27),
nikkei_ind_code VARCHAR(6),
corp_number VARCHAR(13),
etc5 VARCHAR(2)
);
INSERT INTO needs2019uncons.ya01 SELECT * FROM needs2019.ya01 WHERE a09='1'
     AND a10='10' AND a27='0';
```

```
42 : (中略)
43 INSERT INTO needs2019uncons.yl01 SELECT * FROM needs2019.yl01 WHERE a09='1'
       AND a10='10' AND a27='0';
44 INSERT INTO needs2019uncons.firmlist SELECT * FROM needs2019.firmlist;
```

なお，SQL スクリプト17と18の（データベース名以外の）異なっている箇所としては，

(WHERE 句条件 2)　WHERE a09='1' AND a10='10' AND a27 =='0'

であるが，ヘッダーパートの仕様（表 4）から，「連結・単独フラグ」a09 として 1（単独）を選択し，かつ「決算別フラグ」a10 として 10（本決算），さらに，「連結基準フラグ」a27 として 0（単独）を選択している点にあり，この指定によって単独本決算のデータベース needs2019uncons を構築できる．

以上のデータベース構築の流れを簡略化して可視化したものを図21に与える．

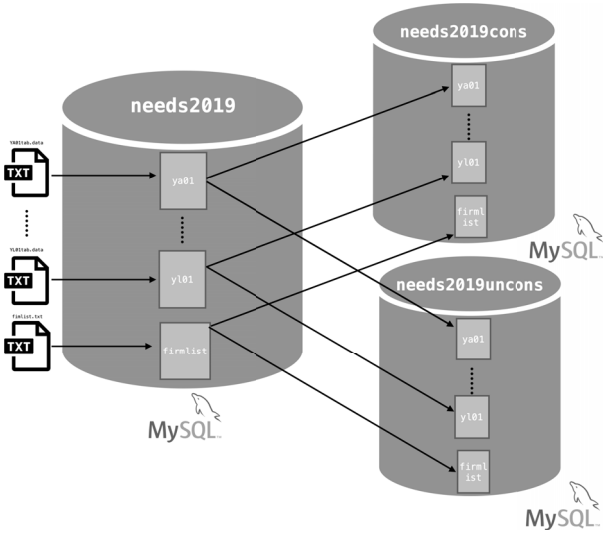


図21：MySQL によるデータベース **needs2019，needs2019cons，needs2019uncons** の構築

なお，この処理はディレクトリ DBMaking（図17参照）をカレントとし，ターミナルで以下のように入力することによって実行できる．

ターゲット MSDB の実行: macOS，Ubuntu 共通

```
% make MSDB
```

macOS 上での，この処理時間は，スクリプト12において，2，9 行目の実行結果を比較することによってわかる．

macOS 上でのターゲット MSDB の処理時間の計測

```
% cat start-MSDB.txt
Fri Sep  4 18:57:45 JST 2020
% cat end-MSDB.txt
Fri Sep 4  19:03:43 JST 2020
```

この結果から， 5 分58秒であることがわかる[32]．

一方，Ubuntu 上で実行した結果も以下に与える．

Ubuntu 上でのターゲット MSDB の処理時間の計測

```
$ cat start-MSDB.txt
Wed Aug 19 16:55:15 JST 2020
$ cat end-MSDB.txt
Wed Aug 19 17:02:46 JST 2020
```

この結果から， 7 分31秒である[33]．

■ PostgreSQL の場合

RDBMS として，PostgreSQL を利用する場合は，macOS と Ubuntu（すなわち，MAPP と LAPP）の間で構築するためのスクリプトをそれぞれ準備する必要があるため，それぞれの場合に分けて述べる[34]．

32）この結果は，MacBook Pro 2018 で計測したものである．
33）この結果は，MacBook Pro 2018 上で VMware によって構築した Ubuntu 18.04 の仮想環境で計測したものである．
34）今回構築した環境は，RDBMS を OS にインストールするために，macOS と Ubuntu 上のパッケージ管理システム（Package Management System: PMS）を，それぞれ，

macOS（MAPP）環境のもとでのデータベース構築　構築手順（NS2），（NS3），（NS4）を実行するスクリプトを Makefile のターゲット PGDB に記述した（スクリプト19参照）．スクリプト19において，2行目と19行目は処理時間を計測するための指定である．

スクリプト19：Makfile：ターゲット PGDB（macOS）

```
1  PGDB:
2        date > start-PGDB.txt
3        /bin/bash replaceloadPostgreSQL-CDIR.sh
4        psql postgres < ./loadtoPostgreSQL.sql
5        /bin/bash replaceloadfirmlistPostgreSQL.sh
6        psql postgres < ./loadfirmlisttoPostgreSQL.sql
7        /bin/bash replacedumpPostgreSQL-CDIR-cons.sh
8        psql postgres < ./dumpfromPostgreSQL-cons.sql
9        /bin/bashreplaceloadPostgreSQL-CDIR-cons.sh
10       psqlpostgres < ./loadtoPostgreSQL-cons.sql
11       /bin/bash replaceloadfirmlistPostgreSQL-cons.sh
12       psql postgres < ./loadfirmlisttoPostgreSQL-cons.sql
13       /bin/bash replacedumpPostgreSQL-CDIR-uncons.sh
14       psql postgres < ./dumpfromPostgreSQL-uncons.sql
15       /bin/bash replaceloadPostgreSQL-CDIR-uncons.sh
16       psql postgres < ./loadtoPostgreSQL-uncons.sql
17       /bin/bash replaceloadfirmlistPostgreSQL-uncons.sh
18       psql postgres < ./loadfirmlisttoPostgreSQL-uncons.sql
19       date > end-PGDB.txt
```

スクリプト19の3，4行目が構築手順（NS2）を実現するためのものであり，5，6行目によって構築手順（NS3）が，さらに7〜18行目の指定で構築手順（NS4）が実現する．

まず，構築手順（NS2）を実現するためのスクリプトについてみる．3行目で利用されているシェルスクリプト replaceloadPostgreSQL-CDIR.sh の内容（スクリプト20）を見ると，2行目でカレントディレクトリの情報を環境変数に代入（CDIR=$PWD）しており，3行目では sed コマンドを利用して，SQL スクリプトファイル loadtoPostgreSQLgen.sql（スクリプト21）における文字列 CDIR をカレントディレクトリの情報で置換（ス

---

brew（Homebrew）と apt を用いて PostgreSQL をインストールしている．データベースを構築するためのスクリプトに差異があるのは，これらの PMS を用いてインストールした際に，RDBMS のルート権限などの付与の仕方が異なっているためである．なお，今回の環境には PostgreSQL は，macOS にはバージョン12.2，Ubuntu には12.4をインストールして運用している．

クリプト21の20〜22行目）し，その結果をSQLスクリプトファイルloadtoPostgreSQL.sqlにリダイレクション（>）機能を使って出力している．この意味で，SQLスクリプトファイルloadtoPostgreSQL.sqlは，シェルスクリプトreplaceloadPostgreSQL-CDIR.shによってSQLスクリプトファイルloadtoPostgreSQLgen.sqlから生成（generate）される（図22参照）．

スクリプト20：replaceloadPostgreSQL-CDIR.sh

```
1 #!/bin/bash
2 CDIR=$PWD
3 sed -e "s|CDIR|$CDIR|g" loadtoPostgreSQLgen.sql > loadtoPostgreSQL.sql
```

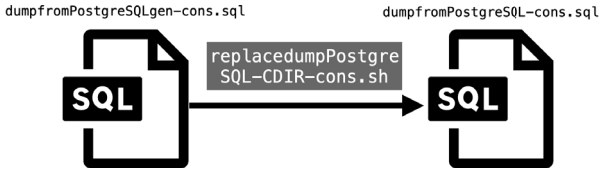


図22：SQLスクリプトファイルloadtoPostgreSQL.sqlの生成

スクリプト21：loadtoPostgreSQLgen.sql（一部抜粋）

```
1  DROP DATABASE IF EXISTS needs2019;
2  CREATE DATABASE needs2019;
3  \c needs2019
4  DROP TABLE IF EXISTS ya01;
5  : (中略)
6  DROP TABLE IF EXISTS yl01;
7  CREATE TABLE ya01 (
8  a01 VARCHAR(4),
9  a02 BIGINT,
10 : (中略)
11 a46 VARCHAR(3),
12 b001 VARCHAR(14),
13 b002 VARCHAR(14),
14 : (中略)
15 b210 VARCHAR(14)
16 );
17 CREATE TABLE yb01 (LIKE ya01);
18 : (中略)
19 CREATE TABLE yl01 (LIKE ya01);
20 COPY public.ya01 FROM 'CDIR/YA01tab.data';
21 : (中略)
22 COPY public.yl01 FROM 'CDIR/YL01tab.data';
```

生成されたSQLスクリプトファイルloadtoPostgreSQL.sqlは，データベースneeds2019と，テーブルya01〜yl01を作成（2〜19行目）した後，データファイルYA01tab.data〜YL01tab.dataから，テーブルya01〜yl01へデータをロードするためのものであり，実際にターゲットPGDB（スクリプト19）の4行目で実行される．

次に，構築手順（NS3）を実現するためのスクリプトとして，Makefileのターゲット PGDB（スクリプト19）の5行目で実行されるスクリプトファイル replaceloadfirmlistPostgreSQL.sh（スクリプト22）は，replaceloadPostgreSQL-CDIR.sh（スクリプト20）と同様の役割を果たす．つまり，2行目でカレントディレクトリの情報を環境変数に代入（CDIR=$PWD）し，3行目でsedコマンドを利用して，SQLスクリプトファイルloadfirmlisttoPostgreSQLgen.sql（スクリプト22）における文字列CDIRをカレントディレクトリの情報で置換（スクリプト23の22行目）し，その結果をSQLスクリプトファイルloadfirmlisttoPostgreSQL.sqlにリダイレクション（>）機能を使って出力している．この仕様から，SQLスクリプトファイルloadfirmlisttoPostgreSQL.sqlは，シェルスクリプトreplaceloadfirmlistPostgreSQL.shによってSQLスクリプトファイルloadfirmlisttoPostgreSQLgen.sqlから生成される（図23参照）．

スクリプト22：replaceloadfirmlistPostgreSQL.sh

```
1 #!/bin/bash
2 CDIR=$PWD
3 sed -e "s|CDIR|$CDIR|g" loadfirmlisttoPostgreSQLgen.sql >
    loadfirmlisttoPostgreSQL.sql
```

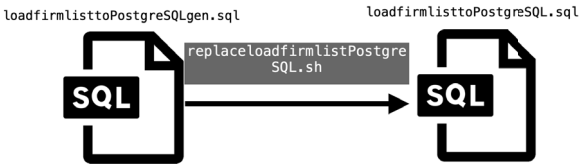


図23：SQLスクリプトファイルloadfirmlisttoPostgreSQL.sqlの生成

スクリプト23：loadfirmlisttoPostgreSQLgen.sql

```
\c needs2019
DROP TABLE IF EXISTS firmlist;
CREATE TABLE firmlist (
record_type VARCHAR(4),
nikkei_corp_code VARCHAR(7),
stock_code VARCHAR(4),
etc1 VARCHAR(1),
etc2 VARCHAR(9),
finance_code VARCHAR(4),
etc3 VARCHAR(11),
firmname VARCHAR(30),
firmname_jp VARCHAR(30),
firmname_jpk VARCHAR(50),
address VARCHAR(80),
zip_code VARCHAR(7),
phone_number VARCHAR(15),
etc4 VARCHAR(27),
nikkei_ind_code VARCHAR(6),
corp_number VARCHAR(13),
etc5 VARCHAR(2)
);
COPY public.firmlist FROM 'CDIR/firmlist.txt';
```

生成されたSQLスクリプトファイル loadfirmlisttoPostgreSQL.sql は，データベース needs2019 においてテーブル firmlist を作成（3〜21行目）した後，収録企業に関する情報が収められたファイル firmlist.txt から，テーブル firmlist へデータをロードするためのものであり，実際にはターゲット PGDB（スクリプト19）の6行目で実行される．

最後に，構築手順（NS4）を実現するためのスクリプトを解説する．ここで，注意を要する事項としては，今回利用したPostgreSQLのバージョンでは，MySQLのようにデータベースからデータベースへテーブルの情報を「流し込む」機能がデフォルトで用意されていないということである．すなわち，新たなデータベースを作るためには，一旦データをデータベースからダンプした後，改めてデータをロードする必要がある．このため，PostgreSQL用のターゲットはMySQLのものに比べて冗長な記述となっている．

Makefile のターゲット PGDB（スクリプト19）における7〜12行目が連結本決算のデータベースを構築するためのものである．まず，7行目で指定されている replacedumpPostgreSQL-CDIR-cons.sh（スクリプト24）

は，replaceloadPostgreSQL-CDIR.sh（スクリプト20）と同様の役割を果たす．つまり，2行目でカレントディレクトリの情報を環境変数に代入（CDIR=$PWD）し，3行目でsedコマンドを利用して，SQLスクリプトファイルdumpfromPostgreSQLgen-cons.sql（スクリプト25）における文字列CDIRをカレントディレクトリの情報で置換（スクリプト25の2〜4行目）し，その結果をSQLスクリプトファイルdumpfromPostgreSQL-cons.sqlにリダイレクション（>）機能を使って出力している．この仕様から，SQLスクリプトファイルdumpfromPostgreSQL-cons.sqlは，シェルスクリプトreplacedumpPostgreSQL-CDIR-cons.shによってSQLスクリプトファイルdumpfromPostgreSQLgen-cons.sqlから生成される（図24参照）．

スクリプト24：replacedumpPostgreSQL-CDIR-cons.sh

```
1 #!/bin/bash
2 CDIR=$PWD
3 sed -e "s|CDIR|$CDIR|g" dumpfromPostgreSQLgen-cons.sql > dumpfromPostgreSQL
    -cons.sql
```

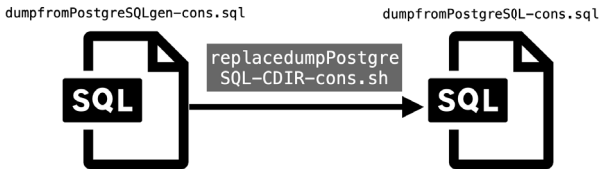


図24：SQLスクリプトファイルdumpfromPostgreSQL-cons.sqlの生成

スクリプト25：dumpfromPostgreSQLgen-cons.sql（一部抜粋）

```
1 \c needs2019
2 \COPY (SELECT * FROM public.ya01 WHERE a09='2' AND a10='10' AND a27!='0')
    TO 'CDIR/YA01tab.cons.data' WITH CSV DELIMITER E'\t';
3 :（中略）
4 \COPY (SELECT * FROM public.yl01 WHERE a09='2' AND a10='10' AND a27!='0')
    TO 'CDIR/YL01tab.cons.data' WITH CSV DELIMITER E'\t';
```

生成されたSQLスクリプトファイルdumpfromPostgreSQL-cons.sql

は，既存のデータベースのテーブル needs2019.ya01 の全ての列（*）から，SQL 問合せに関する WHERE 句に連結本決算を満たす条件（WHERE 句条件 1）参照）を与え，制約付きでレコードを抽出し，カレントディレクトリへタブ区切りファイル YA01tab.cons.data として出力している（2 行目）．これを同様に繰り返し，カレントディレクトリへタブ区切りファイル YL01tab.cons.data として出力している（4 行目）．このスクリプトは，実際にターゲット PGDB（スクリプト19）の 8 行目で実行される．この工程で出力された連結本決算のデータファイル YA01tab.cons.data〜YL01tab.cons.data に基づくデータベースを構築するスクリプトがターゲット PGDB（スクリプト19）の 9，10行目である．9 行目で指定されている replaceloadPostgreSQL-CDIR-cons.sh（スクリプト26）は，replaceloadPostgreSQL-CDIR.sh（スクリプト20）と同様の役割を果たす．つまり，2 行目でカレントディレクトリの情報を環境変数に代入（CDIR=$PWD）し，3 行目で sed コマンドを利用して，SQL スクリプトファイル loadtoPostgreSQLgen-cons.sql（スクリプト27）における文字列 CDIR をカレントディレクトリの情報で置換（スクリプト27の22行目）し，その結果を SQL スクリプトファイル loadtoPostgreSQL-cons.sql にリダイレクション（>）機能を使って出力している．この仕様から，SQL スクリプトファイル loadtoPostgreSQL-cons.sql は，シェルスクリプト replaceloadPostgreSQL-CDIR-cons.sh によって SQL スクリプトファイル loadtoPostgreSQLgen-cons.sql から生成される．

スクリプト26：replaceloadPostgreSQL-CDIR-cons.sh

```
1 #!/bin/bash
2 CDIR=$PWD
3 sed -e "s|CDIR|$CDIR|g" loadtoPostgreSQLgen-cons.sql > loadtoPostgreSQL-
      cons.sql
```

スクリプト27：loadtoPostgreSQLgen-cons.sql（一部抜粋）

```
1 DROP DATABASE IF EXISTS needs2019cons;
2 CREATE DATABASE needs2019cons;
3 \c needs2019cons
```

```
 4 DROP TABLE IF EXISTS ya01;
 5 : (中略)
 6 DROP TABLE IF EXISTS yl01;
 7 CREATE TABLE ya01 (
 8 a01 VARCHAR(4),
 9 a02 BIGINT,
10 : (中略)
11 a46 VARCHAR(3),
12 b001 VARCHAR(14),
13 b002 VARCHAR(14),
14 : (中略)
15 b210 VARCHAR(14)
16 );
17 CREATE TABLE yb01 (LIKE ya01);
18 : (中略)
19 CREATE TABLE yl01 (LIKE ya01);
20 COPY public.ya01 FROM 'CDIR/YA01tab.cons.data';
21 : (中略)
22 COPY public.yl01 FROM 'CDIR/YL01tab.cons.data';
```

生成されたSQLスクリプトファイルloadtoPostgreSQL-cons.sqlは，データベースneeds2019consと，テーブルya01〜yl01を作成（2〜19行目）した後，データファイルYA01tab.cons.data〜YL01tab.cons.dataから，テーブルya01〜yl01へデータをロードするためのものであり，実際にはターゲットPGDB（スクリプト19）の10行目で実行される．連結本決算のデータベースneeds2019consの仕上げとして，収録企業に関する情報が収められたテーブルfirmlistを作成するためのスクリプトについて説明する．ターゲットPGDB（スクリプト19）の11行目で実行されるreplaceloadfirmlistPostgresSQL-cons.sh（スクリプト28）の2行目でカレントディレクトリの情報を環境変数に代入（CDIR=$PWD）し，3行目でsedコマンドを利用して，SQLスクリプトファイルloadfirmlisttoPostgreSQLgen-cons.sql（スクリプト29）における文字列CDIRをカレントディレクトリの情報で置換（スクリプト29の22行目）し，その結果をSQLスクリプトファイルloadfirmlisttoPostgreSQL-cons.sqlにリダイレクション（>）機能を使って出力している．よって，SQLスクリプトファイルloadfirmlisttoPostgreSQL-cons.sqlは，シェルスクリプトreplaceloadfirmlistPostgreSQL.shから生成される。

スクリプト28：replaceloadfirmlistPostgreSQL-cons.sh

```
1 #!/bin/bash
2 CDIR=$PWD
3 sed -e "s|CDIR|$CDIR|g" loadfirmlisttoPostgreSQLgen-cons.sql >
    loadfirmlisttoPostgreSQL-cons.sql
```

スクリプト29：loadfirmlisttoPostgreSQLgen-cons.sql

```
1  \c needs2019cons;
2  DROP TABLE IF EXISTS firmlist;
3  CREATE TABLE firmlist(
4  record_type VARCHAR(4),
5  nikkei_corp_code VARCHAR(7),
6  stock_code VARCHAR(4),
7  etc1 VARCHAR(1),
8  etc2 VARCHAR(9),
9  finance_code VARCHAR(4),
10 etc3 VARCHAR(11),
11 firmname VARCHAR(30),
12 firmname_jp VARCHAR(30),
13 firmname_jpk VARCHAR(50),
14 address VARCHAR(80),
15 zip_code VARCHAR(7),
16 phone_number VARCHAR(15),
17 etc4 VARCHAR(27),
18 nikkei_ind_code VARCHAR(6),
19 corp_number VARCHAR(13),
20 etc5 VARCHAR(2)
21 );
22 \COPY public.firmlist FROM 'CDIR/firmlist.txt';
```

生成されたSQLスクリプトファイルloadfirmlisttoPostgreSQL-cons.sqlは，データベースneeds2019consにおいてテーブルfirmlistを作成（3〜21行目）した後，収録企業に関する情報が収められたファイルfirmlist.txtから，テーブルfirmlistへデータをロードするためのものであり，実際にターゲットPGDB（スクリプト19）の12行目で実行される．

以上の処理（Makefileのターゲット PGDB（スクリプト19）の7〜12行目の実行）によって，連結本決算のデータベースが構築される．ターゲットPGDB（スクリプト19）の残りの処理（13〜18行目）は，単独本決算のデータベースを構築するためのもので，連結本決算のデータベースを構築するための処理との違いは，スクリプト中で文字列consをuconsに置き換えることと，既存のデータベースのテーブルneeds2019.*から単独本決算の

データを抽出する際の条件が（WHERE **句条件** 2)）に変更されている点以外は，本質的には変わらない．よって，冗長となるため説明は割愛する．

以上のデータベース構築の流れを簡略化して可視化したものを図25に与える．

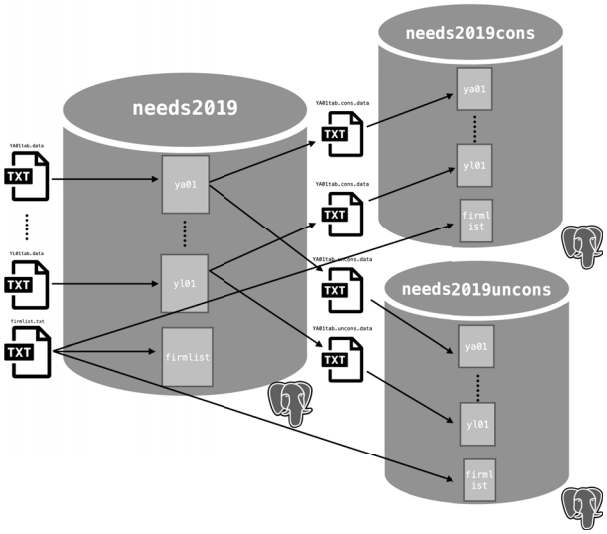


図25：macOS 環境のもとでの PostgreSQL によるデータベース needs2019，needs2019cons，needs2019uncons の構築（MAPP）

図25は macOS 上でのデータベースの構築の流れを表しているが，Ubuntu 上でも全く同じことがいえることに注意が必要である．

なお，この処理はディレクトリ DBMaking（図17参照）において，ターミナルで以下のように入力することによって実行できる．

macOS 上でのターゲット PGDB の実行

```
% make PGDB
```

この処理時間は，スクリプト19において，2，19行目の実行結果を比較することによってわかる．

macOS 上でのターゲット PGDB の処理時間の計測

```
% cat start-PGDB.txt
Sun Sep  6 14:11:42 JST 2020
% cat end-PGDB.txt
Sun Sep  6 14:18:58 JST 2020
```

この結果から，7分16秒であることがわかる[35]．

Ubuntu（LAPP）環境のもとでのデータベース構築　macOS 環境（MAPP）と Ubuntu 環境（LAPP）に対するデータベースを構築するためのスクリプトの唯一の違いは，Makefile におけるターゲット PGDB にある．

スクリプト30：Makfile：ターゲット PGDB（Ubuntu）

```
PGDB:
	date > start-PGDB.txt
	/bin/bash replaceloadPostgreSQL-CDIR.sh
	sudo -u postgres psql < ./loadtoPostgreSQL.sql
	/bin/bash replaceloadfirmlistPostgresSQL.sh
	sudo -u postgres psql < ./loadfirmlisttoPostgreSQL.sql
	/bin/bash replacedumpPostgreSQL-CDIR-cons.sh
	sudo -u postgres psql < ./dumpfromPostgreSQL-cons.sql
	/bin/bash replaceloadPostgreSQL-CDIR-cons.sh
	sudo -u postgres psql < ./loadtoPostgreSQL-cons.sql
	/bin/bash replaceloadfirmlistPostgresSQL-cons.sh
	sudo -u postgres psql < ./loadfirmlisttoPostgreSQL-cons.sql
	/bin/bash replacedumpPostgreSQL-CDIR-uncons.sh
	sudo -u postgres psql < ./dumpfromPostgreSQL-uncons.sql
	/bin/bash replaceloadPostgreSQL-CDIR-uncons.sh
	sudo -u postgres psql < ./loadtoPostgreSQL-uncons.sql
	/bin/bash replaceloadfirmlistPostgresSQL-uncons.sh
	sudo -u postgres psql < ./loadfirmlisttoPostgreSQL-uncons.sql
	date > end-PGDB.txt
```

macOS 用のターゲット PGDB（スクリプト19）と，Ubuntu 用のもの（スクリプト30）を比較することによって，PostgreSQL との対話型インターフェース psql を実行する権限の仕様が若干異なっていることがわかる．これは，macOS と Ubuntu のそれぞれの PMS（brew，apt）でインストールした PostgreSQL のバージョンと仕様に差異があるためである[36]．

35）この結果は，MacBook Pro 2018 で計測したものである．

36）macOS と Ubuntu 用の PostgreSQL の設定を詳細に実行することによって，同様のスクリプトで実行できる可能性もあるが，ここではスクリプトを OS 毎に変更することによって対応した．

ただし，データベースの構築のためには，macOS と同様に，ディレクトリ DBMaking（図17のディレクトリ名 MT-general2019-macOS を MT-general2019-Ubuntu に置き換えたもの）において，Ubuntu のターミナルで以下のように make コマンドを実行すればよい．

Ubuntu 上でのターゲット PGDB の実行

```
$ make PGDB
```

この処理時間は，スクリプト30において，2，19行目の実行結果を比較することによってわかる．

Ubuntu 上でのターゲット PGDB の処理時間の計測

```
$ cat start-PGDB.txt
Wed Aug 19 17:10:54 JST 2020
$ cat end-PGDB.txt
Wed Aug 19 17:16:59 JST 2020
```

この結果から，6 分 5 秒であることがわかる[37]．

付録 C　企業財務データ抽出システム SKWAD

ここでは，企業財務データ抽出システム SKWAD について解説する（トップページは図26を参照）．なお，セキュリティに関する観点から，システムのソースコードを部分的に割愛する．

37）この結果は，MacBook Pro 2018 上で VMware によって構築した Ubuntu 18.04 の仮想環境で計測したものである．

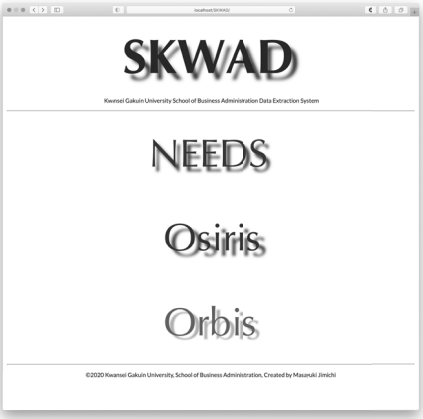


図26：データ抽出システム SKWAD のトップページ

このシステムを構成するファイルのディレクトリ構成（一部抜粋）については図27を参照されたい．

```
SKWAD
├──── NEEDS
│       ├──── 2019
│       └──── index.html
├──── index.html
└──── logo
        ├──── NEEDS.png
        ├──── Orbis.png
        ├──── Osiris.png
        └──── SKWAD.png
```

図27：データ抽出システム SKWAD を構成するファイルのディレクトリ構成（一部抜粋）

ここで，システムのトップページの実体は，図27におけるディレクトリ SKWAD[38] のサブディレクトリにある HTML ファイル index.html であり，このページに表示されるロゴの画像ファイルが logo ディレクトリに格納さ

38）実際のシステム開発では，データが毎年バージョンアップすることを考慮して，SKWAD ディレクトリには，開発年度をディレクトリ名に入れており（たとえば，SKWAD2020），そこから，SKWAD へ ln コマンドによって，リンクを張るような仕様としている．このことによって，年度が変わってもユーザは同一の URL でシステムにアクセスできるようになる．

れている．このトップページの NEEDS ロゴ（アイコンNEEDS）をクリックすることによって，NEEDS 企業財務データのデータベースからデータを抽出するためのトップページ（図28）に移動することができる．

図28：NEEDS 企業財務データを抽出するためのトップページ

なお，このページの実体は，図27におけるサブディレクトリ NEEDS にある HTML ファイル index.html である．ここで，サブディレクトリ NEEDS のさらに下のディレクトリ 2019 の構造を図29に与える．

```
NEEDS
├──── 2019
│   ├──── codes
│   │   ├──── NIKKEIcode.fld
│   │   ├──── NIKKEIcode.htm
│   │   ├──── NIKKEIcode.pdf
│   │   ├──── datainformation.fld
│   │   └──── datainformation.htm
│   ├──── cons
│   │   ├──── NEEDS-general-cons-2019.html
│   │   └──── SQLtreat-general-cons-2019.php
│   ├──── index.html
│   └──── uncons
│       ├──── NEEDS-general-uncons-2019.html
│       └──── SQLtreat-general-uncons-2019.php
└──── index.html
```

図29：NEEDS 企業財務データ（一般事業会社：2019年版）のデータベースからデータを抽出するシステムのディレクトリ構成（一部抜粋）

NEEDS 企業財務データを抽出するためのページ（図28）から，さらにリンク日経 NEEDS 財務データベース（2019年版）をクリックすることによって，NEEDS 企業財務データ（一般事業会社）2019年版のデータベースからデータを抽出するページに移動することができる（図30参照）．このページの実体は，図29におけるディレクトリ 2019 の直下にある index.html ファイルである．

図30：NEEDS 企業財務データ（一般事業会社）2019年版のデータベースからデータを抽出するためのトップページ

このページにおいて，リンク連結本決算とリンク単独本決算をそれぞれ選択することによって，NEEDS 企業財務データ（一般事業会社）2019年版のデータベースからそれぞれの財務データを抽出するページへ移動できる．これらのリンクの実体は，図29のディレクトリ 2019 におけるサブディレクトリ con の NEEDS-general-cons-2019.html と，サブディレクトリ uncon の NEEDS-general-uncons-2019.html である．

また，「データ情報」の一般事業会社のリンク HTML を選択することによって，データの仕様に関するページへ移動することができ，「日経業種コードテーブル」のリンク HTML と PDF をそれぞれ選択することによって，

日経業種コードに関する情報を HTML 形式または PDF ファイルとして参照することができる．これらのリンクの実体は，図29におけるディレクトリ 2019 のサブディレクトリ codes に格納されている．

NEEDS 企業財務データ（一般事業会社）2019年版のデータベースからデータを抽出するためのページ（図30）のリンク連結本決算を選択することによって，「日経 NEEDS 財務データ抽出システム（一般事業会社：2019年版,連結本決算）」のページに移動することができる（図31参照）．

図31：「日経 NEEDS 財務データ抽出システム（一般事業会社：2019年版，連結本決算）」のページ

V 節でも説明したが，このシステムにおいて，スクリプト入力ボックスに SQL 問合せを入力し，Submit ボタンまたは Download CSV ボタンをクリックすることによって，HTML 形式，または CSV 形式でデータを抽出することができる．

このシステムの概要は図32に与えられる．

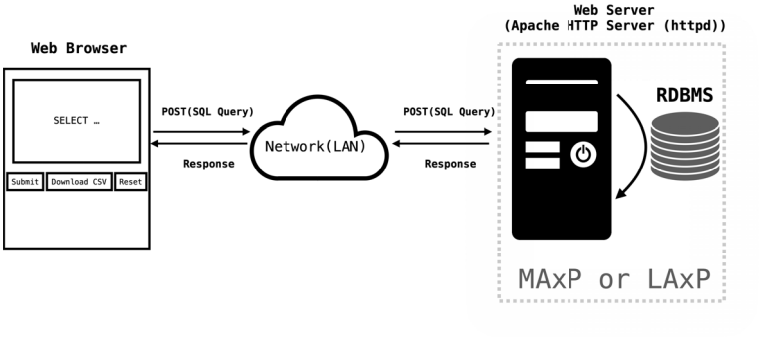


図32：データ抽出システムの概要

図32が示すネットワーク経由でデータを抽出するための処理の流れは以下のようなものである：

データ抽出の流れ

（F-1）Web ブラウザ（クライアント）を利用して，SQL 問合せ（SQL query）を送信
（F-2）サーバが命令を処理，返信
（F-3）抽出結果を受け取り Web ブラウザに表示，またはダウンロード

この処理の流れを実現するための仕様は以下のようなものである：

データ抽出システムの仕様

（SPEC-1）HTML の `FORM` タグを使って SQL 問合せをデータベースサーバへ送信
（SPEC-2）PHP スクリプトを使って，受け取った SQL 問合せ（SQL スクリプト）にもとづきデータを抽出した結果を HTML ファイル，または CSV ファイルとして出力し，返信
（SPEC-3）クライアントが結果の HTML ファイルを受信・表示，または CSV ファイルをダウンロード

ここで仕様（SPEC-1）は，ディレクトリ2019（図29参照）のサブディレクトリ `con` における `NEEDS-general-cons-2019.html` ファイルが，その役割を果たし，実際のコードはスクリプト31で与えられる．

スクリプト31：NEEDS-general-cons-2019.html（一部抜粋）

```
<FORM ACTION='SQLtreat-general-cons-2019.php' METHOD='post'>
  <div align='center'>
      <p>
        <TEXTAREA NAME='SQLcode' ROWS='30' COLS='120'></TEXTAREA>
        <BR>
        <INPUT TYPE='submit' VALUE='Submit' NAME = 'HTML'>
        <INPUT TYPE='submit' VALUE='Download␣CSV' NAME = 'CSV'>
        <INPUT TYPE='reset' VALUE='Reset'>
      </p>
  </div>
</FORM>
```

このスクリプトの役割を以下に与える：

1行目：FORMタグを利用し，属性METHOD='post'と属性ACTION='SQLtreat-general-cons-2019.php'でサーバ上のPHPファイルSQLtreat-general-cons-2019.phpへ入力されたスクリプトを送信する．

4行目：TEXTAREAタグによってSQL問合せを入力するためのボックス（スクロールテキストボックス）を作成し，入力欄の名前を属性NAMEにSQLcodeという文字列を与えており（NAME='SQLcode'），属性ROWSとCOSLを使ってボックスの大きさを30行，120列と指定する（ROWS='30' COLS='120'）．

6行目：INPUTタグの属性TYPE='submit'を利用して，サーバへ提出するための Submit ボタンを作成する．この入力については，NAME = 'HTML'と指定することによって，HTML形式で出力するための名称を設定する．

7行目：INPUTタグの属性TYPE='submit'を利用して，サーバへ提出するための Download CSV ボタンを作成する．この入力については，NAME = 'CSV'と指定することによって，CSV形式で出力するための名称を設定する．

8行目：INPUTタグの属性TYPE='reset'を利用して，入力したスリプトを消去するための Reset ボタンを作成する．

次に，仕様（SPEC-2）は，図29で与えられるディレクトリ 2019 のサブディレクトリ cons の PHP ファイル SQLtreat-general-cons-2019.php が，その役割を果たし，実際のコードはスクリプト32で与えられる．

スクリプト32：SQLtreat-general-cons-2019.php（一部抜粋）

```
 1 <?php
 2 if(isset($_POST['HTML'])) {
 3       $pid = exec('echo␣$$');
 4       $sqlquery = "/tmp/sqlquery-general-2019".$pid.".sql";
 5       $NEEDShtml = "/tmp/NEEDS2019cons".$pid.".html";
 6       $fp = fopen($sqlquery, 'w');
 7       fwrite($fp, $_POST['SQLcode']);
 8       fclose($fp);
 9       system("cat␣$sqlquery␣|␣PGPASSWORD=********␣psql␣-U␣*******␣-d␣
             needs2019cons␣-H␣>␣$NEEDShtml");
10       include($NEEDShtml);
11       exit;}
12 elseif(isset($_POST['CSV'])) {
13       $pid = exec('echo␣$$');
14       $sqlquery = "/tmp/sqlquery-general-2019".$pid.".sql";
15       $NEEDScsv = "/tmp/NEEDS2019cons".$pid.".csv";
16       $fp = fopen($sqlquery, 'w');
17       fwrite($fp, $_POST['SQLcode']);
18       fclose($fp);
19       system("cat␣$sqlquery␣|␣PGPASSWORD=********␣psql␣-U␣*******␣-d␣
             needs2019cons.␣--csv␣>␣$NEEDScsv");
20        header("Content-Type:␣application/octet-stream");
21        header("Content-Disposition:␣attachment;␣filename=\"NEEDS2019cons.
             csv\"");
22        header("Content-Length:␣" . filesize($NEEDScsv));
23        readfile($NEEDScsv);
24       exit;}
25 ?>
```

このスクリプトの役割は以下のようなものである：

Submit ボタンが選択されたときへの対応（2 行目〜11行目）：

2 行目：Submit ボタンが選択（スクリプト31の 6 行目参照）されたかどうかを $_POST['HTML'] がセットされたかどうかで判断する．TRUE（真）の場合は，3 行目以降が実行される．

3 行目：シェルのプロセス ID を取得（exec('echo $$')）し，変数 $pid に代入する．

4 行目：SQL 問合せを格納するスクリプトファイル名をシェルのプロセス

ID 付きで定義し，変数 $sqlquery へ代入する．

5 行目：抽出結果を保存する HTML ファイル名をシェルのプロセス ID 付きで定義し，変数 $NEEDShtml へ代入する．

6 行目：PHP 関数 fopen を利用して，変数 $sqlquery で定義されるファイルを書き込みモード 'w' で開く．

7 行目：スクリプト31の 4 行目で定義されたボックスの入力欄の名前 'SQLcode' から，その入力内容を変数 $sqlquery で定義された SQL スクリプトファイルへ変数 $fp を通じて入力内容を出力する．

8 行目：PHP 関数 fclose を利用して SQL スクリプトファイルを閉じる．

9 行目：変数 $sqlquery で定義された SQL スクリプトファイルの内容を cat で展開後，パイプ機能 (|) を通じて，psql に引き渡し，オプション -d でデータベース needs2019cons を指定して，データを抽出後，オプション -H を指定することによって HTML 形式で変数 $NEEDShtml で指定されたファイルへ出力する．

10行目：PHP 関数 include で結果として出力された HTML ファイルを読み込む．

Download CSV ボタンが選択されたときへの対応（12行目～24行目）

12行目：Submit ボタンが選択（スクリプト31の 7 行目参照）されたかどうかを $_POST['CSV'] がセットされたかどうかで判断する．TRUE（真）の場合は，13行目以降が実行される．

13行目：シェルのプロセス ID を取得（exec('echo $$')）し，変数 $pid に代入する．

14行目：SQL 問合せを格納するスクリプトファイル名をシェルのプロセス ID 付きで定義し，変数 $sqlquery へ代入する．

15行目：抽出結果を保存する CSV ファイル名をシェルのプロセス ID 付きで定義し，変数 $NEEDScsv へ代入する．

16行目：PHP 関数 fopen を利用して，変数 $sqlquery で定義されるファ

イルを書き込みモード 'w' で開く.

17行目：スクリプト31の4行目で定義されたボックスの入力欄の名前 'SQLcode' から，その入力内容を変数 $sqlquery で定義された SQL スクリプトファイルへ変数 $fp を通じて入力内容を出力する.

18行目：PHP 関数 fclose を利用して SQL スクリプトファイルを閉じる.

19行目：変数 $sqlquery で定義された SQL スクリプトファイルの内容を cat で展開後，パイプ機能 (|) を通じて，psql に引き渡し，オプション -d でデータベース needs2019cons を指定して，データを抽出後，オプション -csv を指定することによって CSV 形式で変数 $NEEDScsv で指定されたファイルへ出力する.

20行目：PHP 関数 header で "Content-Type: application/octet-stream" と指定することによって，ファイル形式に関係なくダウンロードを開始する.

21行目：PHP 関数 header で "Content-Disposition: attachment; filename=\"NEEDS2019cons.csv\"" と指定することによって，ファイル名を NEEDS2019cons.csv とする.

22行目：PHP 関数 header で "Content-Length: " . filesize($NEEDScsv) と指定することによって，ファイルサイズを取得し，ダウンロードの進捗を表示する.

23行目：PHP 関数 readfile で CSV ファイルを出力する.

なお，ここで与えたスクリプトには，セキュリティの観点から，PostgreSQL のユーザ名やパスワード，さらに対話型インターフェース psql のパスなどは明記していない．もし，本稿を参考にする際には，これらの情報を適切に設定されたい.